

UNISIM
TMS320C3X Unit tests comprehensive list

Gilles Mouchard
Daniel Gracia Pérez
Reda Nouacer

CEA List

2009, Nov 6

```

Subdirectory: loadstore_int_reg/ldi
Pattern: patterns/2ops_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_dst_int_reg.txt (0 tests)
Random input: loadstore_int_reg/ldi/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldi r1, r2
ldiu st, r3

```

```

Subdirectory: loadstore_int_indir/ldi/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldi **ar2(1), r1          ← instruction under test
ldiu st, r2

```

Subdirectory: loadstore_int_indir/ldi/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 *go after the end of the source buffer*
 ldiu r0, st
 ldi *-ar2(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldi *++ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldi *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldi *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 ldi *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldi *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldi *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldi *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldi *--ar4(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldi *ar4++(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to the source buffer*

addi 15, ar4 *go at the end of the source buffer*

ldiu r1, st

ldi *ar4--(ir0), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

and 4 - 1, r0 *crop random value between 0 and bk - 1*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r1, st

ldi *ar4++(ir0)%, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir0_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir0_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir0_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir0_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldi *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldi *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldi *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir1.add_mod
Pattern: patterns/src_int_indir_preind_ir1.add_mod.dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1.add_mod.dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir1.add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldi *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_preind_ir1.sub_mod
Pattern: patterns/src_int_indir_preind_ir1.sub_mod.dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1.sub_mod.dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_preind_ir1.sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldi *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir1.add_mod
Pattern: patterns/src_int_indir_postind_ir1.add_mod.dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1.add_mod.dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir1.add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldi *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir1 *load ir1 with this random value*
ldiu ar2, ar4 *load a pointer to the source buffer*
addi 15, ar4 *go at the end of the source buffer*
ldiu r1, st
ldi *ar4--(ir1), r2 ← *instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu 4, bk *load block size (should be at most 8)*
and 4 - 1, r0 *crop random value between 0 and bk - 1*
ldiu r0, ir1 *load ir1 with this random value*
ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
addi 7, ar4
andn 7, ar4 *align circular buffer start on block size*
ldiu r1, st
ldi *ar4++(ir1)%, r2 ← *instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir1_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir1_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir1_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir1_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldi *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi/indir
Pattern: patterns/src_int_indir_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldi *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 ldi *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldi/0
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldi/0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldi 0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldi/1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldi/1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldi 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldi/-1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldi/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldi -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldi/-32768
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldi/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldi -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldi/32767
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldi/32767/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldi 32767, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_dir/ldi
Pattern: patterns/src_int_dir_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_dst_int_reg.txt (0 tests)
Random input: loadstore_int_dir/ldi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldiu *ar2, r1 load input value
 sti r1, @_input store input value into _input
 ldi @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldi_ar_update_ordering
Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: loadstore_int_indir/ldi_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldi *ar4++(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: loadstore_int_reg/ldiu

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldiu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldiu r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_indir/ldiu/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiu *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_predisp_sub

Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_indir/ldiu/indir_predisp_sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 go after the end of the source buffer

ldiu r0, st

ldiu *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldiu *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 go after the end of the source buffer
 ldiu r0, st
 ldiu *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldiu *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 15, ar4 *go at the end of the source buffer*

ldiu r0, st

ldiu *ar4--(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

ldiu *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

ldiu *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiu *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiu *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldiu *ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldiu *-ar2(ir0), r2 \leftarrow *instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldiu *++ar4(ir0), r2 \leftarrow *instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiu *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiu *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiu *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldiu *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldiu *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldiu *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiu *--ar4(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiu *ar4++(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiu *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiu *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiu/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiu/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiu *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldiu/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiu/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiu/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiu/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiu/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiu/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiu -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldiu/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiu/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiu -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldiu/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiu/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiu 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldiu

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldiu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldiu @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiu_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldiu_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldilo

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldilo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldilo r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilo *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldilo *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldilo *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldilo *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldilo *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldilo *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilo *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilo *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilo *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilo *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldilo *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldilo *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldilo *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldilo *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilo *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldilo *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldilo *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldilo *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldilo *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldilo *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilo *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldilo *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilo *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilo *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilo/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilo/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilo *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldilo/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilo/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilo 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldilo/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilo/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilo 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldilo/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldilo/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldilo -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldilo/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldilo/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldilo -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldilo/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldilo/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldilo 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldilo

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldilo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldilo @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilo_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldilo_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldilo *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldils

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldils/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldils r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldils *+ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
ldils *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
ldils *++ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldils *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldils *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldils *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldils *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldils *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldils *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldils *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldils *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldils *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldils *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldils *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldils *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldils *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldils *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldils *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldils *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldils *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldils *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldils *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldils *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldils/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldils *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldils/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldils/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldils *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldils/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldils/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldils 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldils/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldils/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldils 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldils/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldils/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldils -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldils/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldils/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldils -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldils/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldils/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldils 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldils

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldils/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldils @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldils_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldils_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldils *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldihi

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldihi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldihi r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihi *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldihi *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldihi *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldihi *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldihi *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldihi *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldihi *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldihi *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldihi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldihi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldihi *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldihi *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldihi *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldihi *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldihi *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldihi *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldihi *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldihi *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldihi *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldihi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldihi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldihi *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihi *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldihi *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldihi/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihi/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihi 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldihi/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihi/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihi 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldihi/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihi/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldihi -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldihi/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihi/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldihi -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldihi/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihi/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldihi 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldihi

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldihi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldihi @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihi_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldihi_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldihi *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldihs

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldihs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldihs r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihs *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldihs *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldihs *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldihs *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldihs *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldihs *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldihs *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldihs *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldihs *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldihs *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldihs *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldihs *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldihs *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldihs *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldihs *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldihs *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldihs *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldihs *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldihs *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldihs *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldihs *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldihs *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldihs *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihs *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldihs/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldihs/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldihs *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldihs/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihs/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihs 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldihs/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihs/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldihs 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldihs/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihs/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldihs -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldihs/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihs/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldihs -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldihs/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldihs/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldihs 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldihs

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldihs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldihs @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldihs_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldihs_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldihs *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldieq

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldieq/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldieq r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldieq *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldieq *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldieq *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldieq *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldieq *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldieq *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldieq *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldieq *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldieq *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldieq *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldieq *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldieq *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldieq *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldieq *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldieq *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldieq *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldieq *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldieq *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldieq *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldieq *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldieq *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldieq *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldieq *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldieq/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldieq *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldieq/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldieq/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldieq *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldieq/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldieq/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldieq 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldieq/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldieq/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldieq 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldieq/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldieq/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldieq -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldieq/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldieq/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldieq -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldieq/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldieq/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldieq 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldieq
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_dir/ldieq/random.txt (100 tests)
Assembly pattern under test:

```
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
.data
_input .word 55555555h      input value
.text
ldiu r0, st
ldiu *ar2, r1               load input value
sti r1, @_input             store input value into _input
ldieq @_input, r2           ← instruction under test
ldiu st, r3
```

Subdirectory: loadstore_int_indir/ldieq_ar_update_ordering
Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: loadstore_int_indir/ldieq_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:

```
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldieq *ar4++(1), ar4       ← instruction under test
ldiu st, r1
```

Subdirectory: loadstore_int_reg/ldine
Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_reg/ldine/random.txt (100 tests)
Assembly pattern under test:

```
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldine r1, r2               ← instruction under test
ldiu st, r3
```

Subdirectory: loadstore_int_indir/ldine/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldine *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldine *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldine *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldine *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldine *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldine *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldine *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldine *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldine *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldine *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldine *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldine *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldine **++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldine *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldine *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldine *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldine *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldine *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldine *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldine *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldine *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldine *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldine *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldine/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldine *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldine/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldine/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldine *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldine/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldine/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldine 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldine/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldine/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldine 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldine/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldine/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldine -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldine/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldine/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldine -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldine/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldine/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldine 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldine

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldine/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldine @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldine_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldine_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldine *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldilt

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldilt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldilt r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilt *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldilt *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldilt *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldilt *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldilt *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldilt *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldilt *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldilt *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilt *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilt *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldilt *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldilt *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldilt *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldilt *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilt *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldilt *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldilt *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldilt *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilt *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldilt *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilt *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldilt *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilt *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilt/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilt/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilt *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldilt/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilt/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilt 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldilt/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilt/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilt 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldilt/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilt/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldilt -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldilt/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilt/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldilt -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldilt/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilt/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldilt 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldilt

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldilt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldilt @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilt_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldilt_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldilt *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldile

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldile/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldile r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldile *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldile *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldile *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldile *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldile *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldile *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldile *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldile *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldile *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldile *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldile *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldile *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldile *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldile *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldile *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldile *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldile *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldile *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldile *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldile *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldile *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldile *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldile *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldile/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldile *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldile/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldile/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldile *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldile/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldile/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldile 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldile/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldile/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldile 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldile/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldile/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldile -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldile/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldile/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldile -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldile/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldile/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldile 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldile

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldile/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldile @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldile_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldile_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldile *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldigt

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldigt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldigt r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldigt *+ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
ldigt *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
ldigt *++ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldigt *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldigt *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldigt *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldigt *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldigt *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldigt *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldigt *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldigt *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldigt *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldigt *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldigt *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldigt *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldigt *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldigt *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldigt *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldigt *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldigt *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldigt *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldigt *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldigt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldigt *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldigt/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldigt/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldigt *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldigt/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldigt/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldigt 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldigt/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldigt/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldigt 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldigt/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldigt/-1/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldigt -1, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_int_imm/ldigt/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldigt/-32768/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldigt -32768, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_int_imm/ldigt/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldigt/32767/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldigt 32767, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_int_dir/ldigt

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldigt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldigt @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldigt_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldigt_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldigt *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldige

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldige/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldige r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldige *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldige *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldige *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldige *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldige *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldige *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldige *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldige *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldige *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldige *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldige *ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldige *-ar2(ir0), r2 \leftarrow *instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldige *++ar4(ir0), r2 \leftarrow *instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldige *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldige *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldige *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldige *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldige *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldige *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldige *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldige *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldige *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldige *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldige/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldige *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldige/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldige/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldige *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldige/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldige/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldige 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldige/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldige/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldige 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldige/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldige/-1/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldige -1, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_int_imm/ldige/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldige/-32768/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldige -32768, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_int_imm/ldige/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldige/32767/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldige 32767, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_int_dir/ldige

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldige/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldige @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldige_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldige_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldige *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldinv

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldinv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldinv r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldinv *+ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
ldinv *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
ldinv *++ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldinv *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldinv *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldinv *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldinv *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldinv *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldinv *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldinv *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldinv *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinv *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinv *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinv *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldinv *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldinv *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldinv *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinv *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinv *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinv *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinv *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinv *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldinv/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinv/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinv 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinv/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinv/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinv 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinv/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinv/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldinv -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldinv/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinv/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldinv -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldinv/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinv/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldinv 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldinv

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldinv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldinv @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinv_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldinv_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldinv *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldiv

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldiv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldiv r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiv **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldiv *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldiv **ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldiv *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldiv *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldiv *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiv *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiv *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldiv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldiv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldiv **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldiv *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldiv *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiv *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiv *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiv *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldiv *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldiv *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldiv *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiv *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiv *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiv *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiv *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiv *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldiv/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiv/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiv 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiv/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiv/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiv 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiv/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiv/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiv -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldiv/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiv/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiv -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldiv/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiv/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiv 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldiv

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldiv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldiv @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiv_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldiv_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiv *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldinuf

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldinuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldinuf r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinuf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldinuf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldinuf ***ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldinuf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldinuf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldinuf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldinuf *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldinuf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldinuf **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldinuf *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldinuf +++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinuf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinuf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinuf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldinuf **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldinuf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinuf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinuf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinuf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinuf *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinuf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinuf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinuf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinuf *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldinuf/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinuf/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinuf 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinuf/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinuf/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinuf 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinuf/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinuf/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinuf -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldinuf/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinuf/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinuf -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldinuf/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinuf/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinuf 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldinuf

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldinuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldinuf @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinuf_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldinuf_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldinuf *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldiuf

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldiuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldiuf r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldiuf *+ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
ldiuf *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
ldiuf *++ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldiuf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldiuf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldiuf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiuf *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiuf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldiuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldiuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldiuf *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, r2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldiuf *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
r2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldiuf *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiuf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiuf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiuf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldiuf *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldiuf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiuf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiuf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiuf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiuf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiuf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiuf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiuf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiuf *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldiuf/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiuf/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiuf 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiuf/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiuf/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiuf 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiuf/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiuf/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldiuf -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldiuf/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiuf/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldiuf -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_imm/ldiuf/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiuf/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldiuf 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_dir/ldiuf

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldiuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldiuf @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiuf_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldiuf_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiuf *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldinlv

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldinlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldinlv r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldinlv **ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
ldinlv **ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
ldinlv ***ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldinlv *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldinlv *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldinlv *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldinlv *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldinlv *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinlv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinlv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldinlv **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldinlv *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldinlv +++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinlv *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinlv *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinlv *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldinlv **ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldinlv *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldinlv +++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinlv *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinlv *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinlv *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinlv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinlv **ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinlv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinlv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinlv *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldinlv/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinlv/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinlv 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinlv/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinlv/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinlv 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinlv/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinlv/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinlv -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldinlv/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinlv/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinlv -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldinlv/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinlv/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinlv 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldinlv

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldinlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldinlv @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinlv_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldinlv_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldinlv *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldilv

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldilv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldilv r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldilv *+ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
ldilv *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
ldilv *++ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldilv *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldilv *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldilv *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldilv *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldilv *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldilv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldilv *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, r2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldilv *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
r2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldilv *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldilv *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilv *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldilv *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldilv *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldilv *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldilv *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldilv *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilv *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldilv *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldilv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilv *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldilv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldilv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldilv *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldilv/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilv/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilv 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldilv/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldilv/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldilv 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldilv/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldilv/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldilv -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldilv/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldilv/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldilv -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldilv/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldilv/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldilv 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldilv

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldilv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldilv @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldilv_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldilv_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldilv *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldinluf

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldinluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldinluf r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinluf *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldinluf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldinluf *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldinluf *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldinluf *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldinluf *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinluf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinluf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinluf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldinluf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldinluf *ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldinluf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinluf *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinluf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinluf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinluf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldinluf **ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldinluf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinluf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldinluf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinluf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldinluf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldinluf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinluf **ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldinluf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldinluf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldinluf *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldinluf/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinluf/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinluf 0, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinluf/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldinluf/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldinluf 1, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldinluf/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinluf/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinluf -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldinluf/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinluf/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinluf -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldinluf/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldinluf/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldinluf 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldinluf

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldinluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldinluf @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldinluf_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldinluf_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldinluf *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldiluf

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldiluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldiluf r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldiluf **ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
ldiluf *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
ldiluf ***ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldiluf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldiluf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldiluf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiluf *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldiluf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldiluf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldiluf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldiluf **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldiluf *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldiluf +++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiluf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiluf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiluf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldiluf **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldiluf *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldiluf *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldiluf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiluf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldiluf *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldiluf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiluf **ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldiluf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldiluf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldiluf *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldiluf/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiluf/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiluf 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiluf/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldiluf/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiluf 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldiluf/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiluf/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiluf -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldiluf/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiluf/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiluf -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldiluf/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldiluf/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiluf 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldiluf

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldiluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldiluf @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldiluf_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldiluf_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiluf *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_reg/ldizuf

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_reg/ldizuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldizuf r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldizuf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldizuf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldizuf ***ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldizuf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldizuf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldizuf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldizuf *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldizuf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldizuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldizuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldizuf **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
ldizuf *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldizuf +++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldizuf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldizuf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldizuf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldizuf **ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

ldizuf *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

ldizuf *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldizuf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldizuf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldizuf *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldizuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldizuf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_indir/ldizuf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_indir/ldizuf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldizuf *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_int_imm/ldizuf/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldizuf/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldizuf 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldizuf/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: loadstore_int_imm/ldizuf/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldizuf 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_int_imm/ldizuf/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldizuf/-1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldizuf -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldizuf/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldizuf/-32768/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldizuf -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_imm/ldizuf/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_imm/ldizuf/32767/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldizuf 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_int_dir/ldizuf

Pattern: patterns/src_int_dir_src_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)

Random input: loadstore_int_dir/ldizuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

ldizuf @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: loadstore_int_indir/ldizuf_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: loadstore_int_indir/ldizuf_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldizuf *ar4++(1), ar4 *← instruction under test*

ldiu st, r1

Subdirectory: loadstore_int_indir/sti/indir_predisp_add

Pattern: patterns/src_int_reg_dst_int_indir_predisp_add.pat

Manually selected input: inputs/src_int_reg_dst_int_indir_predisp_add.txt (0 tests)

Random input: loadstore_int_indir/sti/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

---- OUTPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

sti r0, **ar2(1) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_predisp_sub

Pattern: patterns/src_int_reg_dst_int_indir_predisp_sub.pat

Manually selected input: inputs/src_int_reg_dst_int_indir_predisp_sub.txt (0 tests)

Random input: loadstore_int_indir/sti/indir_predisp_sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

---- OUTPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

addi 16, ar2 *go after the end of the source buffer*

sti r0, *-ar2(1) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_predisp_add_mod
Pattern: patterns/src_int_reg_dst_int_indir_predisp_add_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_predisp_add_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu ar2, ar4
 sti r0, **++ar4(1) ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_predisp_sub_mod
Pattern: patterns/src_int_reg_dst_int_indir_predisp_sub_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_predisp_sub_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu ar2, ar4
 addi 16, ar4 go after the end of the source buffer
 sti r0, *--ar4(1) ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postdisp_add_mod
Pattern: patterns/src_int_reg_dst_int_indir_postdisp_add_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postdisp_add_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu ar2, ar4
 sti r0, *ar4++(1) ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postdisp_sub_mod
Pattern: patterns/src_int_reg_dst_int_indir_postdisp_sub_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postdisp_sub_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 sti r0, *ar4--(1) ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_reg_dst_int_indir_postdisp_add_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postdisp_add_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 sti r0, *ar4++(1)% *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_reg_dst_int_indir_postdisp_sub_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postdisp_sub_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 sti r0, *ar4--(1)% *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_reg_dst_int_indir_postdisp_add_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postdisp_add_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 sti r0, *ar4++(1)% *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_reg_dst_int_indir_postdisp_sub_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postdisp_sub_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 sti r0, *ar4--(1)% *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir0_add
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir0_add.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir0_add.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 sti r1, *ar2(ir0) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir0_sub
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir0_sub.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir0_sub.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 sti r1, *-ar2(ir0) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir0_add_mod
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir0_add_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir0_add_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 sti r1, **ar4(ir0) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir0_sub_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir0_sub_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 sti r1, *--ar4(ir0) ← *instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postind_ir0_add_mod
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir0_add_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir0_add_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 sti r1, *ar4++(ir0) ← *instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir0_sub_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir0_sub_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 sti r1, *ar4--(ir0) ← *instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir0_add_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir0_add_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 sti r1, *ar4++(ir0)% ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir0_sub_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir0_sub_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 sti r1, *ar4--(ir0)% ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir0_add_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir0_add_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 sti r1, *ar4++(ir0)% ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postind_ir0_sub_circ_mod_bk.5
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir0_sub_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir0_sub_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir0_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 sti r1, *ar4--(ir0)% *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir1_add
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir1_add.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir1_add.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 sti r1, **ar2(ir1) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir1_sub
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir1_sub.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir1_sub.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 sti r1, *-ar2(ir1) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir1_add_mod
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir1_add_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir1_add_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 sti r1, *++ar4(ir1) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_reg_dst_int_indir_preind_ir1_sub_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_preind_ir1_sub_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 sti r1, *--ar4(ir1) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postind_ir1_add_mod
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir1_add_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir1_add_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 sti r1, *ar4++(ir1) *← instruction under test*

Subdirectory: loadstore_int_indir/sti/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir1_sub_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir1_sub_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:

```

---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
and 15, r0                crop random value between 0 and 15
ldiu r0, ir1              load ir1 with this random value
ldiu ar2, ar4             load a pointer to the source buffer
addi 15, ar4              go at the end of the source buffer
sti r1, *ar4--(ir1)       ← instruction under test

```

Subdirectory: loadstore_int_indir/sti/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir1_add_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir1_add_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

```

---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
ldiu 4, bk                load block size (should be at most 8)
and 4 - 1, r0             crop random value between 0 and bk - 1
ldiu r0, ir1              load ir1 with this random value
ldiu ar2, ar4             load a pointer to a buffer of 16 words
addi 7, ar4
andn 7, ar4               align circular buffer start on block size
sti r1, *ar4++(ir1)%      ← instruction under test

```

Subdirectory: loadstore_int_indir/sti/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir1_sub_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir1_sub_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

```

---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
ldiu 4, bk                load block size (should be at most 8)
and 4 - 1, r0             crop random value between 0 and bk - 1
ldiu r0, ir1              load ir1 with this random value
ldiu ar2, ar4             load a pointer to a buffer of 16 words
addi 7, ar4
andn 7, ar4               align circular buffer start on block size
sti r1, *ar4--(ir1)%      ← instruction under test

```

Subdirectory: loadstore_int_indir/sti/indir_postind_ir1_add_circ_mod_bk.5
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir1_add_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir1_add_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir1_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 sti r1, *ar4++(ir1)% ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postind_ir1_sub_circ_mod_bk.5
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir1_sub_circ_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir1_sub_circ_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir1_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 sti r1, *ar4--(ir1)% ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir
Pattern: patterns/src_int_reg_dst_int_indir.pat
Manually selected input: inputs/src_int_reg_dst_int_indir.txt (0 tests)
Random input: loadstore_int_indir/sti/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 sti r0, *ar2 ← instruction under test

Subdirectory: loadstore_int_indir/sti/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_reg_dst_int_indir_postind_ir0_add_bit_rev_mod.pat
Manually selected input: inputs/src_int_reg_dst_int_indir_postind_ir0_add_bit_rev_mod.txt (0 tests)
Random input: loadstore_int_indir/sti/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 sti r1, *ar4++(ir0)b *← instruction under test*

Subdirectory: loadstore_int_dir/sti
Pattern: patterns/src_int_reg_dst_int_dir.pat
Manually selected input: inputs/src_int_reg_dst_int_dir.txt (0 tests)
Random input: loadstore_int_dir/sti/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 .data
 _output .word 55555555h *output value*
 .text
 ldiu r0, st
 sti r1, @_output *← instruction under test*
 ldiu st, r2
 ldiu @_output, r3 *load output value from _output*
 sti r3, *ar2 *store output value*

Subdirectory: loadstore_int_indir/sti_ar_update_ordering
Pattern: patterns/2ops_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: loadstore_int_indir/sti_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 sti ar2, *ar2++(1) *← instruction under test*
 ldiu ar2, ar5
 ldiu st, r1

Subdirectory: loadstore_float_reg/lde

Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_reg/lde/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r2: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu r0, st

lde r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_predisp_add

Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/lde/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

lde *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_predisp_sub

Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/lde/indir_predisp_sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

addi 16, ar2 go after the end of the source buffer

ldiu r0, st

lde *-ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 lde *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 go after the end of the source buffer
 ldiu r0, st
 lde *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 lde *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 lde *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 lde *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 lde *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 lde *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 lde *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 lde *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 lde *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lde *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 lde *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lde *ar4++(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 lde *ar4--(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 lde *+ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 lde *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lde *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 lde *--ar4(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lde *ar4++(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 lde *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lde *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/lde/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/lde/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 lde *+ar2, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/lde/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/lde/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
lde *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/lde/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/lde/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lde 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/lde/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/lde/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lde 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/lde/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/lde/-1.0/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

lde -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/lde/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/lde/1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

lde 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/lde/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/lde/-1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

lde -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/lde/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/lde/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lde 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/lde/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/lde/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lde 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/lde/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/lde/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lde -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/lde/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/lde/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 lde -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/lde
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/lde/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 lde @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldm
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldm/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldm r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldm *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldm *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldm *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldm *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldm *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldm *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldm *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldm *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldm *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldm *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldm *+ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldm *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldm *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldm *--ar4(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldm *ar4++(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldm *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldm *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldm *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldm *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldm *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldm *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldm *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldm *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldm *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldm *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldm *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldm *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldm *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldm *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldm *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldm/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldm/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldm *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldm/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldm/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldm *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldm/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldm/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldm 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldm/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldm 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldm/-1.0/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldm -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldm/1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldm 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldm/-1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldm -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldm/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldm 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldm/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldm 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldm/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldm -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldm/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldm/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldm -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldm
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldm/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldm @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldf
Pattern: patterns/2ops_src_float_reg_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldf *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldf *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldf *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldf *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldf *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldf *+ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldf *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldf *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldf *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldf *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldf *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldf *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldf/indir
Pattern: patterns/src_float_indir_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf *+ar2, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldf *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_imm/ldf/0.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/0.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf 0.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf 1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/-1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/-1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf -1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf 1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/-1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldf -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/2.5594e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldf -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldf/-2.56e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldf
Pattern: patterns/src_float_dir_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfu
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfu/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfu r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ld fu **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ld fu *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ld fu **ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ld fu *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ld fu *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ld fu *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfu *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfu *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfu *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfu *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldfu **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldfu *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 ldfu **ar4(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfu *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfu *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfu *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfu *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfu *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfu *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfu *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfu *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ld fu *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ld fu **ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfu *--ar4(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfu *ar4++(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfu *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfu *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfu *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfu *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfu *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfu/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfu *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfu/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfu/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfu *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfu/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfu/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfu 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfu/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfu 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfu/-1.0/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfu -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfu/1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfu 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfu/-1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfu -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfu/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfu 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfu/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfu 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfu/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfu -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfu/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfu/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ld fu -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfu
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfu/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ld fu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ld fu @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldflo
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldflo/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ld flo r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflo *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldflo *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldflo *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldflo *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldflo *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldflo *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflo *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflo *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldflo *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldflo *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfl *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfl *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfl *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldflo *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflo *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldflo *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflo *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflo *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflo *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfllo *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfllo *+ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfl0 *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfl0 *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldflo *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflo *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfllo *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfllo *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfl *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfl *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfllo *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflo/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfllo *+ar2, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflo/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldflo/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldflo *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldflo/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflo -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflo/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflo/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfl -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldflo
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldflo/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfl *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfl @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfls
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfls/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfls r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfls *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfls *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfls *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfls *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfls *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfls *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfls *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfls *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfls *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfls *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfls *+ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfls *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfls *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfls *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfls *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfls *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfls *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfls *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfls *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfls *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldfls *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfls *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfls *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfls *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfls *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfls *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfls *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfls *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfls *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfls *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfls/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfls *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfls/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfls/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfls *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfls/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfls 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfls 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/-1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfls -1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfls 1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/-1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfls -1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfls 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfls 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfls -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfls/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfls/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfls -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfls
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfls/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ld fu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfls @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfhi
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfhi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhi r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhi *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfhi *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfhi *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfhi *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfhi *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfhi *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfhi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfhi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfhi *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfhi *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldfhi *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldfhi *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 ldfhi *++ar4(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfhi *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfhi *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfhi *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhi *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfhi *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfhi *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfhi *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfhi *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfhi *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfhi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfhi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfhi *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhi/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhi *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfhi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfhi *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfhi/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhi 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhi 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/-1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhi -1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhi 1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/-1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhi -1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhi 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhi 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhi -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhi/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhi/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhi -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfhi
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfhi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfhi @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfhs
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfhs/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhs r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhs *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfhs *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfhs *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfhs *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfhs *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfhs *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfhs *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfhs *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfhs *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfhs *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfhs *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfhs *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfhs *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfhs *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfhs *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfhs *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfhs *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfhs *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfhs *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfhs *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfhs *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfhs *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfhs *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfhs/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhs *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfhs/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfhs/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfhs *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfhs/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfhs -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfhs/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfhs/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfhs -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfhs
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfhs/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfhs @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfeq
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfeq/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfeq r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfeq *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfeq *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfeq *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfeq *--ar4(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfeq *ar4++(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfeq *ar4--(1), r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfeq *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfeq *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfeq *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfeq *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfeq *+ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfeq *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfeq *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfeq *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfeq *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfeq *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfeq *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfeq *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfeq *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfeq *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfeq *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfeq *-ar2(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfeq *++ar4(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfeq *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfeq *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfeq *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfeq *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfeq *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfeq *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfeq *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfeq/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfeq *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfeq/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfeq/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfeq *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfeq/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfeq -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfeq/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfeq/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfeq -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfeq
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfeq/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ld fu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfeq @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfne
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfne/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfne r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfne *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfne *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfne *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_predisp_sub_mod

Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldfne/indir_predisp_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

ldfne *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_postdisp_add_mod

Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldfne/indir_postdisp_add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

ldfne *ar4++(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_postdisp_sub_mod

Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldfne/indir_postdisp_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 15, ar4 *go at the end of the source buffer*

ldiu r0, st

ldfne *ar4--(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfne *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfne *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfne *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfne *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfne *+ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfne *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfne *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfne *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfne *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfne *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfne *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfne *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfne *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfne *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfne *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfne *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfne *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfne *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfne *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfne *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfne *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfne *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfne *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfne *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfne/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfne *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfne/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfne/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfne *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfne/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfne 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfne 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfne -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfne 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfne -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfne 2.5594e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfne 7.8125e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfne -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfne/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfne/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfne -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfne
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfne/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfne @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldflt
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldflt/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflt r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflt *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldflt *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldflt *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldflt *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldflt *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldflt *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflt *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflt *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflt *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflt *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldflt *+ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldflt *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflt *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldflt *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflt *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldflt *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflt *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflt *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflt *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflt *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldflt *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldflt *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflt *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldflt *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflt *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldflt *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflt *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflt *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflt/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflt *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflt/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldflt/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldflt *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldflt/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflt -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflt/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflt/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflt -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldflt
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldflt/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ld fu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldflt @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfle
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfle/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ld fle r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfl *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfl *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfl *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfl *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfl *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfl *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfl *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfl *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfl *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfl *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfl *ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfl *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfl *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfl *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfl *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfl *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfl *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfl *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfl *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflle *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldflle *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfl *ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfl *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfl *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfl *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfl *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfl *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfile *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfile *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfl *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfle/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfl *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfle/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfle/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfle *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfle/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfle -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfle/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfle/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfl -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfle
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfle/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfr *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfl @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfgt
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfgt/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfgt r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldftg *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldftg *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldftg *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldftg *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldftg *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldftg *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfgt *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfgt *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldftg *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldftg *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfgt *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfgt *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfgt *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldftg *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldftg *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfgt *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfgt *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfgt *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfgt *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfgt *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfgt *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldftg *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldftg *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldftg *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldftg *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldftg *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldftg *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfgt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfgt *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfgt *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfgt/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfgt *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfgt/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfgt/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfgt *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfgt/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfgt 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfgt/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfgt 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfgt/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfgt -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfgt/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfgt 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfgt/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfgt -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfgt/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldIU r0, st
 ldFGT 2.5594e2, r1 ← instruction under test
 ldIU st, r2

Subdirectory: loadstore_float_imm/ldfgt/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldIU r0, st
 ldFGT 7.8125e-3, r1 ← instruction under test
 ldIU st, r2

Subdirectory: loadstore_float_imm/ldfgt/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldIU r0, st
 ldFGT -7.8163e-3, r1 ← instruction under test
 ldIU st, r2

Subdirectory: loadstore_float_imm/ldfgt/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfgt/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfgt -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfgt
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfgt/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfgt @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfge
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfge/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfge r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfge *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfge *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfge *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfge *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfge *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfge *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfge *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfge *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfge *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfge *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfge *+ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfge *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfge *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfge *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfge *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfge *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfge *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfge *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfge *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfge *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfge *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfge *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfge *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfge *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfge *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfge *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfge *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfge *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfge *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfge *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfge/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfge *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfge/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfge/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfge *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfge/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfge -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfge/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfge/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfge -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfge
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfge/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfge @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfnv
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfnv/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnv r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnnv *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfnnv *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfnnv *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfnnv *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfnnv *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfnnv *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnnv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnnv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnnv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnnv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnv/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldlnv *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnv/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldlnv *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnv/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 ldlnv *++ar4(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfnnv *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnnv *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfnnv *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnnv *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnnv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnnv *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnnv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldfnnv *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfnnv *-ar2(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnnv *++ar4(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfnnv *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnnv *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfnnv *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnnv *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnnv *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnnv *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnnv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnnv/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnnv *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfnnv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfnnv *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfnnv/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnnv/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnnv 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnnv/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnnv/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnnv 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnnv/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfnnv/-1.0/random.txt (1 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- *OUTPUTS* ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfnnv -1.0, r1 ← *instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnnv/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfnnv/1.5/random.txt (1 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- *OUTPUTS* ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfnnv 1.5, r1 ← *instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnnv/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfnnv/-1.5/random.txt (1 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- *OUTPUTS* ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfnnv -1.5, r1 ← *instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnnv/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnnv/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnnv 2.5594e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnnv/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnnv/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnnv 7.8125e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnnv/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnnv/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnnv -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnn/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnn/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnn -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfnn
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfnn/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfnn @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfv
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfv/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfv r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ld fv **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ld fv *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ld fv **++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ld fv *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ld fv *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ld fv *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfv **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfv *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfv **ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfv *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfv *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfv *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfv *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfv *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ld fv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ld fv **ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfv *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfv **ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ld fv *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ld fv *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfv *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfv *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfv *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfv/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfv **ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfv *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfv/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfv/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfv 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfv/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfv 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfv/-1.0/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfv -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfv/1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfv 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfv/-1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfv -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfv/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfv 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfv/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfv 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfv/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfv -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfv/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfv/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfv -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfv
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfv/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfv @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfnuf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfnuf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnuf r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnuf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfnuf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfnuf ***ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfnuf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfnuf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfnuf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfnuf *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfnuf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfnuf **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfnuf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnuf ***ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfnuf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnuf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfnuf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnuf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnuf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnuf *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfnuf **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfnuf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnuf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfnuf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnuf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfnuf *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnuf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnuf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnuf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnuf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnuf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfnuf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
ldfnuf *ar4++(ir0)b, r2 ← *instruction under test*
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfnuf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf 0.0, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf 1.0, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnuf -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnuf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnuf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnuf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfnuf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfnuf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfnuf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfuf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfuf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfuf r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ld fuf *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ld fuf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ld fuf *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_predisp_sub_mod

Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldfuf/indir_predisp_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

ldfuf *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_postdisp_add_mod

Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldfuf/indir_postdisp_add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

ldfuf *ar4++(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_postdisp_sub_mod

Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldfuf/indir_postdisp_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 15, ar4 *go at the end of the source buffer*

ldiu r0, st

ldfuf *ar4--(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ld fuf *ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ld fuf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ld fuf *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfuf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfuf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfuf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfuf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfuf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfuf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfuf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfuf *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfuf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfuf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfuf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfuf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfuf *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfuf *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfuf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfuf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfuf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfuf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfuf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfuf *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfuf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfuf 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfuf 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/-1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfuf -1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfuf 1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/-1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfuf -1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfuf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfuf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfuf -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfuf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfuf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfuf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfuf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfuf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfuf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfnlv
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfnlv/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnlv r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnlv **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfnlv *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfnlv ***ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfnlv *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfnlv *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfnlv *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnlv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnlv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnlv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnlv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfnlv **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfnlv *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnlv **++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfnlv *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnlv *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfnlv *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnlv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnlv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnlv *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnlv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldfnlv **ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfnlv *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnlv *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfnlv *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnlv *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfnlv *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnlv *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnlv *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfnlv *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfnlv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnlv/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfnlv *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfnlv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfnlv *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfnlv/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfnlv -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfnlv/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfnlv/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldIU r0, st
 ldfnlv -2.56e2, r1 ← instruction under test
 ldIU st, r2

Subdirectory: loadstore_float_dir/ldfnlv
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfnlv/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldIU r0, st
 ldFU *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfnlv @_input, r2 ← instruction under test
 ldIU st, r3

Subdirectory: loadstore_float_reg/ldflv
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldflv/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldIU r0, st
 ldflv r1, r2 ← instruction under test
 ldIU st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflv *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldflv *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldflv *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_predisp_sub_mod

Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldflv/indir_predisp_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

ldflv *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_postdisp_add_mod

Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldflv/indir_postdisp_add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

ldflv *ar4++(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_postdisp_sub_mod

Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_indir/ldflv/indir_postdisp_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 15, ar4 *go at the end of the source buffer*

ldiu r0, st

ldflv *ar4--(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflv *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldflv *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldflv *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldflv *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldflv *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldflv *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 ldflv *++ar4(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldflv *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflv *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldflv *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflv *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflv *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflv *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldflv *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldflv *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflv *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldflv *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldflv *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldflv *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflv *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflv *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldflv *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldflv *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldflv/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflv *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldflv/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldflv/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldflv *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldflv/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflv 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflv/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflv 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflv/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/-1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldIU r0, st
 ldflv -1.0, r1 ← instruction under test
 ldIU st, r2

Subdirectory: loadstore_float_imm/ldflv/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldIU r0, st
 ldflv 1.5, r1 ← instruction under test
 ldIU st, r2

Subdirectory: loadstore_float_imm/ldflv/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/-1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldIU r0, st
 ldflv -1.5, r1 ← instruction under test
 ldIU st, r2

Subdirectory: loadstore_float_imm/ldflv/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflv 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflv/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflv 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflv/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldflv -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldflv/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldflv/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldflv -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldflv
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldflv/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ld fu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldflv @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfnluf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfnluf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ld fnluf r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfmluf *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfmluf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfmluf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfmluf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfmluf *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfntluf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfntluf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfntluf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfntluf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfntluf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfntluf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnluf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnluf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnluf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnluf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnluf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnluf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnluf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfnluf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfnluf/indir_preind_ir0.add
Pattern: patterns/src_float_indir_preind_ir0.add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_preind_ir0.add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldfnluf *ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnluf/indir_preind_ir0.sub
Pattern: patterns/src_float_indir_preind_ir0.sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_preind_ir0.sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ldfnluf *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnluf/indir_preind_ir0.add.mod
Pattern: patterns/src_float_indir_preind_ir0.add.mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.add.mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_preind_ir0.add.mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 ldfnluf *++ar4(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfntluf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfntluf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfmluf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfmluf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir_postind_ir0_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postind_ir0_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfntluf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir_postind_ir0_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postind_ir0_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfntluf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_postind_ir0_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_postind_ir0_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfmluf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfmluf **ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnluf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfnluf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfnluf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfnluf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfnluf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfmluf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfmluf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfmluf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfmluf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfmluf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfmluf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir_postind_ir1_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postind_ir1_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfntluf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir_postind_ir1_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postind_ir1_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfntluf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir_postind_ir1_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postind_ir1_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfntluf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfntluf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfntluf **ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfntluf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfntluf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfntluf *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfntluf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfntluf/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfntluf 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfntluf/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfntluf/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfntluf 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfmluf/-1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfmluf/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- *OUTPUTS* ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfmluf -1.0, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfmluf/1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfmluf/1.5/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- *OUTPUTS* ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfmluf 1.5, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfmluf/-1.5
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfmluf/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- *OUTPUTS* ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfmluf -1.5, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfmluf/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfmluf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- *OUTPUTS* ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfmluf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfmluf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfmluf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- *OUTPUTS* ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfmluf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfmluf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfmluf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- *OUTPUTS* ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfmluf -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfmluf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfmluf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfmluf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfmluf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfmluf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfmluf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfluf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfluf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfluf r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfluf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfluf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfluf ***ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfluf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfluf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfluf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfluf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfluf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfluf *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 ldfluf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfluf **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfluf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfluf **++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfluf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfluf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfluf *ar4--(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfluf *ar4++(ir0)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfluf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfluf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfluf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 ldfluf **ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfluf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfluf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfluf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfluf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfluf *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfluf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfluf *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfluf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfluf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfluf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfluf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfluf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfluf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfluf *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfluf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfluf/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfluf 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfluf/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfluf 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfluf/-1.0/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfluf -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfluf/1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfluf 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfluf/-1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfluf -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfluf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfluf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfluf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfluf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfluf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfluf -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfluf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfluf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfluf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfluf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfluf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ld fu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfluf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_reg/ldfzuf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_reg/ldfzuf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ld fzuf r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfzuf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 ldfzuf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfzuf ***ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 ldfzuf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ldfzuf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 ldfzuf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfzuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfzuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfzuf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ldfzuf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfzuf **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 ldfzuf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfzuf **++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfzuf *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfzuf *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfzuf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfzuf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfzuf *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfzuf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfzuf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfzuf **ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ldfzuf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfzuf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ldfzuf *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ldfzuf *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ldfzuf *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfzuf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfzuf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 ldfzuf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ldfzuf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: loadstore_float_indir/ldfzuf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfzuf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: loadstore_float_indir/ldfzuf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
ldfzuf *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: loadstore_float_imm/ldfzuf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfzuf/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfzuf 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfzuf/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfzuf 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfzuf/-1.0/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfzuf -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfzuf/1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfzuf 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: loadstore_float_imm/ldfzuf/-1.5/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldfzuf -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfzuf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfzuf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfzuf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfzuf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfzuf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldfzuf -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: loadstore_float_imm/ldfzuf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_imm/ldfzuf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ldfzuf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: loadstore_float_dir/ldfzuf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: loadstore_float_dir/ldfzuf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 ldfzuf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: loadstore_float_indir/stf/indir_predisp_add
Pattern: patterns/src_float_reg_dst_float_indir_predisp_add.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_predisp_add.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 stf r0, **ar2(1) ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_predisp_sub
Pattern: patterns/src_float_reg_dst_float_indir_predisp_sub.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_predisp_sub.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 addi 16, ar2 go after the end of the source buffer
 stf r0, *-ar2(1) ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_predisp_add_mod
Pattern: patterns/src_float_reg_dst_float_indir_predisp_add_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_predisp_add_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu ar2, ar4
 stf r0, *++ar4(1) ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_predisp_sub_mod
Pattern: patterns/src_float_reg_dst_float_indir_predisp_sub_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_predisp_sub_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu ar2, ar4
 addi 16, ar4 go after the end of the source buffer
 stf r0, *--ar4(1) ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postdisp_add_mod
Pattern: patterns/src_float_reg_dst_float_indir_postdisp_add_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postdisp_add_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu ar2, ar4
 stf r0, *ar4++(1) ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postdisp_sub_mod
Pattern: patterns/src_float_reg_dst_float_indir_postdisp_sub_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postdisp_sub_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 stf r0, *ar4--(1) ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_reg_dst_float_indir_postdisp_add_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postdisp_add_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r0, *ar4++(1)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_reg_dst_float_indir_postdisp_sub_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postdisp_sub_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r0, *ar4--(1)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_reg_dst_float_indir_postdisp_add_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postdisp_add_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r0, *ar4++(1)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_reg_dst_float_indir_postdisp_sub_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postdisp_sub_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 stf r0, *ar4--(1)% *← instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_preind_ir0_add
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir0_add.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir0_add.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 stf r1, *ar2(ir0) *← instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_preind_ir0_sub
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir0_sub.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir0_sub.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 stf r1, *ar2(ir0) *← instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir0_add_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir0_add_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 stf r1, **ar4(ir0) *← instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir0_sub_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir0_sub_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 stf r1, *--ar4(ir0) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir0_add_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir0_add_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 stf r1, *ar4++(ir0) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir0_sub_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir0_sub_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 stf r1, *ar4--(ir0) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir0_add_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir0_add_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r1, *ar4++(ir0)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir0_sub_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir0_sub_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r1, *ar4--(ir0)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir0_add_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir0_add_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r1, *ar4++(ir0)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir0_sub_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir0_sub_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r1, *ar4--(ir0)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_preind_ir1_add
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir1_add.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir1_add.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 stf r1, **ar2(ir1) ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_preind_ir1_sub
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir1_sub.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir1_sub.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 stf r1, *-ar2(ir1) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir1_add_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir1_add_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 stf r1, *++ar4(ir1) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_reg_dst_float_indir_preind_ir1_sub_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_preind_ir1_sub_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 stf r1, *--ar4(ir1) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir1_add_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir1_add_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 stf r1, *ar4++(ir1) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir1_sub_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir1_sub_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 stf r1, *ar4--(ir1) ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir1_add_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir1_add_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 stf r1, *ar4++(ir1)% ← *instruction under test*

Subdirectory: loadstore_float_indir/stf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir1_sub_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir1_sub_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r1, *ar4--(ir1)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir1_add_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir1_add_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r1, *ar4++(ir1)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir1_sub_circ_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir1_sub_circ_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 stf r1, *ar4--(ir1)% ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir
Pattern: patterns/src_float_reg_dst_float_indir.pat
Manually selected input: inputs/src_float_reg_dst_float_indir.txt (0 tests)
Random input: loadstore_float_indir/stf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 stf r0, *ar2 ← instruction under test

Subdirectory: loadstore_float_indir/stf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_reg_dst_float_indir_postind_ir0_add_bit_rev_mod.pat
Manually selected input: inputs/src_float_reg_dst_float_indir_postind_ir0_add_bit_rev_mod.txt (0 tests)
Random input: loadstore_float_indir/stf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
ar4: an auxiliary register
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 stf r1, *ar4++(ir0)b ← instruction under test

Subdirectory: loadstore_float_dir/stf

Pattern: patterns/src_float_reg_dst_float_dir.pat

Manually selected input: inputs/src_float_reg_dst_float_dir.txt (0 tests)

Random input: loadstore_float_dir/stf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r2: a 32-bit integer register (value of st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

.data

_output .word 55555555h *output value*

.text

ldiu r0, st

stf r1, @_output *← instruction under test*

ldiu st, r2

ldfu @_output, r3 *load output value from _output*

stf r3, *ar2 *store output value*

Subdirectory: control_disp/bu

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bu unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bu

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bu r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bud

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bud/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bud unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bud

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bud/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bud r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbu

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbu ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbu

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbu ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbud

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbud/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbud ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbud

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbud/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbud ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callu

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callu unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callu

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callu r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsu

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsu/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsu *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/blo

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/blo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

blo unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/blo

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/blo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

blo r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/blod

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/blod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

blod unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/blod

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/blod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

blod r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dblo

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dblo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dblo ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dblo

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dblo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dblo ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dblod

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dblod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dblod ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dblod

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dblod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dblod ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/calllo

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/calllo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

calllo unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/calllo

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/calllo/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

calllo r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsls

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsls *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bls

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bls unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bls

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bls r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/blsd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/blsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

blsd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/blsd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/blsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

blsd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbls

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbls ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbls

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbls ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dblsd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dblsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dblsd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dblsd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dblsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dblsd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callls

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callls unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callls

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callls r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsls

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsls/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsls *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bhi

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bhi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bhi unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bhi

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bhi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bhi r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bhid

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bhid/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bhid unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bhid

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bhid/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bhid r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbhi

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbhi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbhi ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbhi

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbhi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbhi ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbhid

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbhid/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbhid ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbhid

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbhid/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbhid ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callhi

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callhi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callhi unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callhi

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callhi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callhi r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retshi

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retshi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retshi *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bhs

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bhs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bhs unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bhs

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bhs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bhs r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bhsd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bhsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bhsd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bhsd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bhsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bhsd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbhs

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbhs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbhs ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbhs

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbhs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbhs ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbhsd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbhsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbhsd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbhdsd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbhdsd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbhdsd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callhs

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callhs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callhs unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callhs

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callhs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callhs r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retshs

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retshs/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retshs *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/beq

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/beq/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

beq unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/beq

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/beq/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

beq r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/beqd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/beqd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

beqd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/beqd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/beqd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

beqd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbeq

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbeq/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbeq ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbeq

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbeq/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbeq ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbeqd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbeqd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbeqd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbeqd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbeqd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbeqd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/calleg

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/calleg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

calleg unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/calleg

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/calleg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

calleg r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retseq

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retseq/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retseq *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bne

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bne/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bne unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bne

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bne/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bne r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bned

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bned/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bned unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bned

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bned/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bned r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbne

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbne/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbne ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbne

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbne/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbne ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbned

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbned/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbned ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbned

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbned/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbned ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callne

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callne/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callne unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callne

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callne/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callne r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsne

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsne/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsne *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/blt

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/blt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

blt unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/blt

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/blt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

blt r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bltd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bltd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bltd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bltd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bltd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bltd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dblt

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dblt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dblt ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dblt

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dblt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dblt ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbltd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbltd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbltd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbltd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbltd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbltd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/calllt

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/calllt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

calllt unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/calllt

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/calllt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

calllt r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retslt

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retslt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retslt *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/ble

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/ble/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ble unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/ble

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/ble/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

ble r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bled

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bled/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bled unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bled

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bled/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bled r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dble

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dble/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbld ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dble

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dble/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbld ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbld

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbld/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbld ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbled

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbled/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbled ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callle

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callle/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callle unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callle

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callle/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callle r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsle

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsle/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsle *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bgt

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bgt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bgt unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bgt

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bgt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bgt r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bgtd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bgtd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bgtd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bgtd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bgtd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bgtd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbgt

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbgt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbgt ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbgt

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbgt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbgt ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbgtd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbgtd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbgtd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbgtd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbgtd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbgtd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callgt

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callgt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callgt unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callgt

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callgt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callgt r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsgt

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsgt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsgt *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bge

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bge/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bge unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bge

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bge/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bge r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bged

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bged/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bged unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bged

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bged/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bged r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbge

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbge/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbge ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbge

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbge/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbge ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbged

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbged/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbged ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbged

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbged/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbged ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callge

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callge/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callge unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callge

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callge/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callge r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsge

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsge/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsge *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bnv

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bnv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bnv unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bnv

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bnv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bnv r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bnvd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bnvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bnvd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bnvd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bnvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bnvd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnv

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbnv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbnv ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnv

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbnv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbnv ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnvd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbnvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbnvd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnvd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbnvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbnvd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callnv

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callnv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callnv unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callnv

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callnv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callnv r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsnv

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsnv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsnv *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bv

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bv unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bv

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bv r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bvd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bvd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bvd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bvd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbv

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbv ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbv

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbv ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbvd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbvd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbvd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbvd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callv

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callv unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callv

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callv r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsv

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsv *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bnuf

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bnuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bnuf unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bnuf

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bnuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bnuf r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bnufd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bnufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bnufd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bnufd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bnufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bnufd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnuf

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbnuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbnuf ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnuf

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbnuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbnuf ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnufd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbnufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbnufd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnufd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbnufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbnufd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callnuf

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callnuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callnuf unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callnuf

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callnuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callnuf r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsnuf

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsnuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsnuf *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/buf

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

buf unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/buf

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

buf r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bufd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bufd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bufd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bufd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbuf

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbuf ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbuf

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbuf ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbufd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbufd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbufd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbufd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/calluf

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/calluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

calluf unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/calluf

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/calluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

calluf r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsuf

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsuf *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bnlv

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bnlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bnlv unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bnlv

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bnlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bnlv r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bnlvd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bnlvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bnlvd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bnlvd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bnlvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bnlvd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnlv

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbnlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbnlv ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnlv

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbnlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbnlv ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnlvd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbnlvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbnlvd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnlvd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbnlvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbnlvd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callnlv

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callnlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callnlv unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callnlv

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callnlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callnlv r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsnlv

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsnlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsnlv *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/blv

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/blv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

blv unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/blv

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/blv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

blv r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/blvd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/blvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

blvd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/blvd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/blvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

blvd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dblv

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dblv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dblv ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dblv

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dblv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dblv ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dblvd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dblvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dblvd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dblvd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dblvd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dblvd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/calllv

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/calllv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

calllv unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/calllv

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/calllv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

calllv r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retslv

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retslv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retslv *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bnluf

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bnluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bnluf unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bnluf

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bnluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bnluf r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bnlufd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bnlufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bnlufd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bnlufd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bnlufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bnlufd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnluf

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbnluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbnluf ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnluf

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbnluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbnluf ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbnlufd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbnlufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbnlufd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbnlufd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbnlufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbnlufd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callnluf

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callnluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callnluf unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callnluf

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callnluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callnluf r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsnluf

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsnluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsnluf *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bluf

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bluf unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bluf

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bluf r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/blufd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/blufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

blufd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/blufd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/blufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

blufd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbluf

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbluf ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbluf

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbluf ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dblufd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dblufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dblufd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dblufd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dblufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dblufd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callluf

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callluf unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callluf

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callluf r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retsluf

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retsluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retsluf *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_disp/bzuf

Pattern: patterns/cond_branch_disp.pat

Manually selected input: inputs/cond_branch_disp.txt (0 tests)

Random input: control_disp/bzuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

bzuf unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 into the destination register*

unit_test_end:

Subdirectory: control_reg/bzuf

Pattern: patterns/cond_branch_reg.pat

Manually selected input: inputs/cond_branch_reg.txt (0 tests)

Random input: control_reg/bzuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

bzuf r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/bzufd

Pattern: patterns/cond_delayed_branch_disp.pat

Manually selected input: inputs/cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/bzufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r3: a 32-bit integer register

r5: a 32-bit integer register

r7: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero in destination register*

bzufd unit_test_end *← instruction under test*

ldiu r2, r3 *1st instruction after delayed branch*

ldiu r4, r5 *2nd instruction after delayed branch*

ldiu r6, r7 *3rd instruction after delayed branch*

addi 1, r1 *increment the destination register*

unit_test_end:

Subdirectory: control_reg/bzufd

Pattern: patterns/cond_delayed_branch_reg.pat

Manually selected input: inputs/cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/bzufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5 *load address of branch target*

bzufd r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbzuf

Pattern: patterns/dec_cond_branch_disp.pat

Manually selected input: inputs/dec_cond_branch_disp.txt (0 tests)

Random input: control_disp/dbzuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

dbzuf ar2, unit_test_end *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbzuf

Pattern: patterns/dec_cond_branch_reg.pat

Manually selected input: inputs/dec_cond_branch_reg.txt (0 tests)

Random input: control_reg/dbzuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_end_addr, r2 *load address of branch target*

dbzuf ar2, r2 *← instruction under test*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/dbzufd

Pattern: patterns/dec_cond_delayed_branch_disp.pat

Manually selected input: inputs/dec_cond_delayed_branch_disp.txt (0 tests)

Random input: control_disp/dbzufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

dbzufd ar2, unit_test_end *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_reg/dbzufd

Pattern: patterns/dec_cond_delayed_branch_reg.pat

Manually selected input: inputs/dec_cond_delayed_branch_reg.txt (0 tests)

Random input: control_reg/dbzufd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

r4: a 32-bit integer register

.data

unit_test_end_addr .word unit_test_end *end of assembly pattern address*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu 0, r2 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r4 *put zero into the destination register of 3rd instruction after delayed branch*

ldiu @unit_test_end_addr, r5

dbzufd ar2, r5 *← instruction under test*

ldiu 1, r2 *put 1 in destination register of 1st instruction after delayed branch*

ldiu 2, r3 *put 1 in destination register of 2nd instruction after delayed branch*

ldiu 3, r4 *put 1 in destination register of 3rd instruction after delayed branch*

ldiu 1, r1 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control_disp/callzuf

Pattern: patterns/cond_call_disp.pat

Manually selected input: inputs/cond_call_disp.txt (0 tests)

Random input: control_disp/callzuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

callzuf unit_test_fn *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_reg/callzuf

Pattern: patterns/cond_call_reg.pat

Manually selected input: inputs/cond_call_reg.txt (0 tests)

Random input: control_reg/callzuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

unit_test_fn_addr .word unit_test_fn *address of target function*

.text

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

ldiu @unit_test_fn_addr, r2 *load address of target function*

callzuf r2 *← instruction under test*

ldiu 1, r1 *not taken call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *taken call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/retszuf

Pattern: patterns/cond_return.pat

Manually selected input: inputs/cond_return.txt (0 tests)

Random input: control_abs/retszuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

call unit_test_fn *call target function*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 0, r1 *put zero into the destination register*

retszuf *← instruction under test*

ldiu 1, r1 *not taken return: put 1 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_abs/br

Pattern: patterns/uncond_branch.pat

Manually selected input: inputs/uncond_branch.txt (0 tests)

Random input: control_abs/br/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

---- OUTPUTS ----

r0: a 32-bit integer register

ldiu 0, r0 *put zero into the destination register*

br unit_test_end *← instruction under test*

ldiu 1, r0 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control.abs/brd

Pattern: patterns/uncond_delayed_branch.pat

Manually selected input: inputs/uncond_delayed_branch.txt (0 tests)

Random input: control.abs/brd/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

---- OUTPUTS ----

r0: a 32-bit integer register

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register

ldiu 0, r0 *put zero into the destination register*

ldiu 0, r1 *put zero into the destination register of 1st instruction after delayed branch*

ldiu 0, r2 *put zero into the destination register of 2nd instruction after delayed branch*

ldiu 0, r3 *put zero into the destination register of 3rd instruction after delayed branch*

brd unit_test_end *← instruction under test*

ldiu 1, r1 *1st instruction after delayed branch*

ldiu 2, r2 *2nd instruction after delayed branch*

ldiu 3, r3 *3rd instruction after delayed branch*

ldiu 1, r0 *not taken branch: put 1 in the destination register*

unit_test_end:

Subdirectory: control.abs/call

Pattern: patterns/call.pat

Manually selected input: inputs/call.txt (0 tests)

Random input: control.abs/call/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1 *put zero into the destination register*

call unit_test_fn *← instruction under test*

ldiu 1, r1 *not performed call: put 1 into the destination register*

br unit_test_end *go to the end of assembly pattern*

unit_test_fn:

ldiu 2, r1 *performed call: put 2 into the destination register*

retsu *return from call*

unit_test_end:

Subdirectory: control_imm/trapu/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/traplo/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/traplo/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

traplo 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapls/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapls/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapls 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/traphi/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/traphi/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

traphi 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/traphs/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/traphs/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

traphs 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapeq/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapeq/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapeq 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapne/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapne/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapne 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/traplt/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/traplt/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word _isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

traplt 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/traple/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/traple/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

traple 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapgt/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapgt/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapgt 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapge/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapge/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapge 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapnv/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapnv/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapnv 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapv/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapv/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapv 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapnuf/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapnuf/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapnuf 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapuf/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapuf/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapuf 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapnlv/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapnlv/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapnlv 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/traplv/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/traplv/0/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

traplv 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapnluf/0
Pattern: patterns/cond_trap.pat
Manually selected input: inputs/cond_trap.txt (0 tests)
Random input: control_imm/trapnluf/0/random.txt (100 tests)
Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapnluf 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapluf/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapluf/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word _isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapluf 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapzuf/0

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapzuf/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 0, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapzuf 0 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/1

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/1/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 1, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 1 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/2

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/2/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 2, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 2 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/3

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/3/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 3, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 3 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/4

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/4/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 4, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 4 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/5

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 5, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 5 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/6

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/6/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 6, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 6 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/7

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/7/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 7, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 7 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/8

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/8/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 8, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 8 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/9

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/9/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 9, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 9 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/10

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/10/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 10, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 10 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/11

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/11/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 11, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 11 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/12

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/12/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 12, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 12 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/13

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/13/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 13, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 13 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/14

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/14/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 14, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 14 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/15

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/15/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 15, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 15 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/16

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/16/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 16, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 16 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/17

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/17/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 17, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 17 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/18

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/18/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 18, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 18 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/19

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/19/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 19, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 19 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/20

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/20/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 20, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 20 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/21

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/21/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 21, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 21 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/22

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/22/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 22, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 22 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/23

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/23/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 23, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 23 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/24

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/24/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 24, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 24 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/25

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/25/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 25, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 25 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/26

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/26/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 26, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 26 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_imm/trapu/27

Pattern: patterns/cond_trap.pat

Manually selected input: inputs/cond_trap.txt (0 tests)

Random input: control_imm/trapu/27/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu 0, r1 *Put zero into the destination register*

ldiu 27, ar2 *load ISR number*

ldiu @trap_table_addr, ir0 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *+ar2(ir0) *install br instruction into the trap table*

trapu 27 *trap*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 1, r1 *taken trap: put 1 into the destination register*

reti *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retiu

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retiu/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retiu *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retilo

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retilo/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retilo *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retils

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retils/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retils *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retihi

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retihi/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retihi *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retihs

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retihs/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retihs *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retieq

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retieq/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retieq *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retine

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retine/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retine *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retilt

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retilt/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word _isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retilt *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retil

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retil/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retil *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retigt

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retigt/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retigt *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retige

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retige/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retige *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retinv

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retinv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retinv *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retiv

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retiv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retiv *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retinuf

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retinuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retinuf *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retiuf

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retiuf/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retiuf *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retinlv

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retinlv/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retinlv *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retilv

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retilv/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retilv *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retinluf

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retinluf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retinluf *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retiluf

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retiluf/random.txt (100 tests)

Assembly pattern under test:

---- *INPUTS* ----

r0: a 32-bit integer register (value for st)

---- *OUTPUTS* ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- *microcomputer mode is assumed.*

- *C31 boot loader (and its ISR table) must be mapped to address 0x0.*

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retiluf *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_abs/retizuf

Pattern: patterns/cond_reti.pat

Manually selected input: inputs/cond_reti.txt (0 tests)

Random input: control_abs/retizuf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

.data

isr_addr .word isr *interrupt service routine address*

trap_table_addr .word 809FE0h *address of trap table*

Note:

- microcomputer mode is assumed.

- C31 boot loader (and its ISR table) must be mapped to address 0x0.

br_opcode .word 60000000h *opcode of an unconditional branch*

.text

ldiu r0, st

ldiu @trap_table_addr, ar2 *load address of trap table*

ldiu @isr_addr, ar4 *load ISR address*

or @br_opcode, ar4 *assemble a br instruction branching to ISR*

sti ar4, *ar2 *install br instruction into the trap table (trap #0)*

trapu 0 *trap #0*

br unit_test_end *go to the end of assembly pattern*

_isr:

ldiu 0, r1 *Put zero into the destination register*

retizuf *return from interrupt*

ldiu 1, r1 *Not taken return from interrupt: put 1 into the destination register*

retiu *return from interrupt*

unit_test_end:

Subdirectory: control_reg/rpts

Pattern: patterns/rpts_reg.pat

Manually selected input: inputs/rpts_reg.txt (0 tests)

Random input: control_reg/rpts/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1

and 65535, r2 *crop count to 0-65535*

rpts r2

addi 1, r1 *count the number of times instruction is executed*

Subdirectory: control_dir/rpts

Pattern: patterns/rpts_dir.pat

Manually selected input: inputs/rpts_dir.txt (0 tests)

Random input: control_dir/rpts/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

.data

_rpts_count .word 0h *rpts count*

.text

ldiu r0, st

ldiu 0, r1

ldiu *ar2, r2

and 65535, r2

sti r2, @_rpts_count

rpts @_rpts_count

addi 1, r1 *count the number of times instruction is executed*

Subdirectory: control_indir/rpts

Pattern: patterns/rpts_indir.pat

Manually selected input: inputs/rpts_indir.txt (0 tests)

Random input: control_indir/rpts/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1

ldiu *ar2, r2

and 65535, r2 *crop count to 0-65535*

sti r2, *ar2

rpts *ar2

addi 1, r1 *count the number of times instruction is executed*

Subdirectory: control_imm/rpts

Pattern: patterns/rpts_imm.pat

Manually selected input: inputs/rpts_imm.txt (0 tests)

Random input: control_imm/rpts/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1

rpts 10

addi 1, r1 *count the number of times instruction is executed*

Subdirectory: control_abs/rptb

Pattern: patterns/rptb.pat

Manually selected input: inputs/rptb.txt (0 tests)

Random input: control_abs/rptb/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

ldiu r0, st

ldiu 0, r1

and 65535, r2

ldiu r2, rc

rptb _unit_test_end

addi 1, r1 *count the number of times instruction is executed*

_unit_test_end:

Subdirectory: 2ops_int_reg/absi

Pattern: patterns/2ops_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/absi/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

absi r1, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/absi/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

absi ***ar2(1)*, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_predisp_sub

Pattern: patterns/src_int_indir_predisp_sub_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_sub_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/absi/indir_predisp_sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

absi *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_predisp_add_mod

Pattern: patterns/src_int_indir_predisp_add_mod_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_mod_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/absi/indir_predisp_add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

absi ++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_predisp_sub_mod

Pattern: patterns/src_int_indir_predisp_sub_mod_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_sub_mod_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/absi/indir_predisp_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go at the end of the source buffer*

ldiu r0, st

absi *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 absi *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 absi *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 absi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 absi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 absi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 absi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 absi **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 absi *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 absi **ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 absi *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 absi *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to the source buffer*

addi 15, ar4 *go at the end of the source buffer*

ldiu r1, st

absi *ar4--(ir0), r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

and 4 - 1, r0 *crop random value between 0 and bk - 1*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r1, st

absi *ar4++(ir0)%, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absi *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 absi *+ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 absi *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 absi *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 absi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 absi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 absi *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absi *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absi *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/absi/indir
Pattern: patterns/src_int_indir_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absi *+ar2, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/absi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/absi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 absi *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/absi
Pattern: patterns/src_int_dir_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/absi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 absi @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/absi/0
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/absi/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absi 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/absi/1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/absi/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absi 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/absi/-1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/absi/-1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absi -1, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/absi/-32768
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/absi/-32768/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absi -32768, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/absi/32767
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/absi/32767/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absi 32767, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_reg/negb
Pattern: patterns/2ops_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/negb/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 32-bit integer register
 ---- OUTPUTS ----
 r2: a 32-bit integer register
 r3: a 32-bit integer register (value of st)
 ldiu r0, st
 negb r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_predisp.add

Pattern: patterns/src_int_indir_predisp.add_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp.add_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/negb/indir_predisp.add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

negb *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_predisp.sub

Pattern: patterns/src_int_indir_predisp.sub_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp.sub_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/negb/indir_predisp.sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 go after the end of the source buffer

ldiu r0, st

negb *-ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_predisp.add_mod

Pattern: patterns/src_int_indir_predisp.add_mod_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp.add_mod_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/negb/indir_predisp.add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

negb *+ar4(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go at the end of the source buffer*
 ldiu r0, st
 negb *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 negb *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 negb *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negb *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negb *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negb *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 negb *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 negb *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 negb *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negb *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 negb *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negb *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to the source buffer*

addi 15, ar4 *go at the end of the source buffer*

ldiu r1, st

negb *ar4--(ir0), r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

and 4 - 1, r0 *crop random value between 0 and bk - 1*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r1, st

negb *ar4++(ir0)%, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negb *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 negb *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 negb *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negb *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 negb *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negb *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the source buffer*

addi 15, ar4 *go at the end of the source buffer*

ldiu r1, st

negb *ar4--(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

and 4 - 1, r0 *crop random value between 0 and bk - 1*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r1, st

negb *ar4++(ir1)%, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negb *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negb *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 negb *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/negb/indir
Pattern: patterns/src_int_indir_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negb *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/negb/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negb/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 negb *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_dir/negb

Pattern: patterns/src_int_dir_dst_int_reg.pat

Manually selected input: inputs/src_int_dir_dst_int_reg.txt (0 tests)

Random input: 2ops_int_dir/negb/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

negb @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_imm/negb/0

Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/negb/0/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

negb 0, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_imm/negb/1

Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/negb/1/random.txt (1 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

negb 1, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_imm/negb/-1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negb/-1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negb -1, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/negb/-32768
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negb/-32768/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negb -32768, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/negb/32767
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negb/32767/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negb 32767, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_reg/negi
Pattern: patterns/2ops_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/negi/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 32-bit integer register
 ---- OUTPUTS ----
 r2: a 32-bit integer register
 r3: a 32-bit integer register (value of st)
 ldiu r0, st
 negi r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_predisp.add

Pattern: patterns/src_int_indir_predisp.add_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp.add_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/negi/indir_predisp.add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

negi **+ar2(1)*, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_predisp.sub

Pattern: patterns/src_int_indir_predisp.sub_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp.sub_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/negi/indir_predisp.sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 go after the end of the source buffer

ldiu r0, st

negi **-ar2(1)*, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_predisp.add_mod

Pattern: patterns/src_int_indir_predisp.add_mod_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp.add_mod_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/negi/indir_predisp.add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

negi **+ar4(1)*, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go at the end of the source buffer*
 ldiu r0, st
 negi *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 negi *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 negi *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 negi *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 negi *ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 negi *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negi *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 negi *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negi *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 negi *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 negi *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 negi *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negi *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 negi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 negi *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negi *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/negi/indir
Pattern: patterns/src_int_indir_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negi *+ar2, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/negi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/negi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negi *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/negi
Pattern: patterns/src_int_dir_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/negi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 negi @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/negi/0
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negi/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negi 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/negi/1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negi/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negi 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/negi/-1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negi/-1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negi -1, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/negi/-32768
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negi/-32768/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negi -32768, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/negi/32767
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/negi/32767/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 32-bit integer register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negi 32767, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_reg/not
Pattern: patterns/2ops_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/not/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 32-bit integer register
 ---- OUTPUTS ----
 r2: a 32-bit integer register
 r3: a 32-bit integer register (value of st)
 ldiu r0, st
 not r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/not/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

not *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_predisp_sub

Pattern: patterns/src_int_indir_predisp_sub_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_sub_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/not/indir_predisp_sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 go after the end of the source buffer

ldiu r0, st

not *-ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_predisp_add_mod

Pattern: patterns/src_int_indir_predisp_add_mod_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_mod_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/not/indir_predisp_add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

not *++ar4(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go at the end of the source buffer*
 ldiu r0, st
 not *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 not *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 not *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 not *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 not *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 not *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 not *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 not *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 not *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 not *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 not *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 not *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir0_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to the source buffer*

addi 15, ar4 *go at the end of the source buffer*

ldiu r1, st

not *ar4--(ir0), r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir0_add_circ_mod_bk_4

Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/not/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

and 4 - 1, r0 *crop random value between 0 and bk - 1*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r1, st

not *ar4++(ir0)%, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 not *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 not *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 not *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 not *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 not *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 not *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 not *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 not *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir1_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 crop random value between 0 and 15

ldiu r0, ir1 load ir1 with this random value

ldiu ar2, ar4 load a pointer to the source buffer

addi 15, ar4 go at the end of the source buffer

ldiu r1, st

not *ar4--(ir1), r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu 4, bk load block size (should be at most 8)

and 4 - 1, r0 crop random value between 0 and bk - 1

ldiu r0, ir1 load ir1 with this random value

ldiu ar2, ar4 load a pointer to a buffer of 16 words

addi 7, ar4

andn 7, ar4 align circular buffer start on block size

ldiu r1, st

not *ar4++(ir1)%, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 not *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 not *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 not *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/not/indir
Pattern: patterns/src_int_indir_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 not *+ar2, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/not/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/not/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 not *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/not
Pattern: patterns/src_int_dir_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/not/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 not @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/not/0
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/not/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 not 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/not/1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/not/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 not 1, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/not/-1
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/not/-1/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
not -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/not/-32768
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/not/-32768/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
not -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/not/32767
Pattern: patterns/2ops_src_int_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/not/32767/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
not 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_reg/cmpi
Pattern: patterns/2ops_src_int_reg_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_src_int_reg.txt (0 tests)
Random input: 2ops_int_reg/cmpi/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
ldiu r0, st
cmpi r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpi *+ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
cmpi *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
cmpi *++ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 cmpi *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 cmpi *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 cmpi *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 cmpi *ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 cmpi *ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpi *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 cmpi *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpi *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the source buffer*
addi 15, ar4 *go at the end of the source buffer*
ldiu r1, st
cmpi *ar4--(ir0), r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
ldiu 4, bk *load block size (should be at most 8)*
and 4 - 1, r0 *crop random value between 0 and bk - 1*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
addi 7, ar4
andn 7, ar4 *align circular buffer start on block size*
ldiu r1, st
cmpi *ar4++(ir0)%, r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 cmpi *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 cmpi *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpi *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 cmpi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir1 *load ir1 with this random value*
ldiu ar2, ar4 *load a pointer to the source buffer*
addi 15, ar4 *go at the end of the source buffer*
ldiu r1, st
cmpi *ar4--(ir1), r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
ldiu 4, bk *load block size (should be at most 8)*
and 4 - 1, r0 *crop random value between 0 and bk - 1*
ldiu r0, ir1 *load ir1 with this random value*
ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
addi 7, ar4
andn 7, ar4 *align circular buffer start on block size*
ldiu r1, st
cmpi *ar4++(ir1)%, r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/cmpi/indir
Pattern: patterns/src_int_indir_src_int_reg.pat
Manually selected input: inputs/src_int_indir_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpi *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/cmpi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/cmpi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpi *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/cmpi

Pattern: patterns/src_int_dir_src_int_reg.pat

Manually selected input: inputs/src_int_dir_src_int_reg.txt (0 tests)

Random input: 2ops_int_dir/cmpi/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

cmpi @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_imm/cmpi/0

Pattern: patterns/2ops_src_int_imm_src_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)

Random input: 2ops_int_imm/cmpi/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register (value of st)

ldiu r0, st

cmpi 0, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_imm/cmpi/1

Pattern: patterns/2ops_src_int_imm_src_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)

Random input: 2ops_int_imm/cmpi/1/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register (value of st)

ldiu r0, st

cmpi 1, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_imm/cmpi/-1
Pattern: patterns/2ops_src_int_imm_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)
Random input: 2ops_int_imm/cmpi/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpi -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/cmpi/-32768
Pattern: patterns/2ops_src_int_imm_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)
Random input: 2ops_int_imm/cmpi/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpi -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/cmpi/32767
Pattern: patterns/2ops_src_int_imm_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)
Random input: 2ops_int_imm/cmpi/32767/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpi 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_reg/tstb
Pattern: patterns/2ops_src_int_reg_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_src_int_reg.txt (0 tests)
Random input: 2ops_int_reg/tstb/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
ldiu r0, st
tstb r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
tstb **ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
addi 16, ar2 go after the end of the source buffer
ldiu r0, st
tstb *-ar2(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
ldiu ar2, ar4
ldiu r0, st
tstb **ar4(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 tstb *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 tstb *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 tstb *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_postdisp_add_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postdisp_add_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 tstb *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_postdisp_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 tstb *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 tstb *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 tstb *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 tstb *ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 tstb *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 tstb *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 tstb *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 tstb *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod.src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod.src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the source buffer*
addi 15, ar4 *go at the end of the source buffer*
ldiu r1, st
tstb *ar4--(ir0), r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod.src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod.src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
ldiu 4, bk *load block size (should be at most 8)*
and 4 - 1, r0 *crop random value between 0 and bk - 1*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
addi 7, ar4
andn 7, ar4 *align circular buffer start on block size*
ldiu r1, st
tstb *ar4++(ir0)%, r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 tstb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 tstb *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 tstb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 tstb *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 tstb *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 tstb *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 tstb *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 tstb *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir1 *load ir1 with this random value*
ldiu ar2, ar4 *load a pointer to the source buffer*
addi 15, ar4 *go at the end of the source buffer*
ldiu r1, st
tstb *ar4--(ir1), r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
ldiu 4, bk *load block size (should be at most 8)*
and 4 - 1, r0 *crop random value between 0 and bk - 1*
ldiu r0, ir1 *load ir1 with this random value*
ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
addi 7, ar4
andn 7, ar4 *align circular buffer start on block size*
ldiu r1, st
tstb *ar4++(ir1)%, r2 ← *instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 tstb *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 tstb *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 tstb *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/tstb/indir
Pattern: patterns/src_int_indir_src_int_reg.pat
Manually selected input: inputs/src_int_indir_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 tstb *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/tstb/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_int_reg.txt (0 tests)
Random input: 2ops_int_indir/tstb/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 tstb *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/tstb

Pattern: patterns/src_int_dir_src_int_reg.pat

Manually selected input: inputs/src_int_dir_src_int_reg.txt (0 tests)

Random input: 2ops_int_dir/tstb/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register (value of st)

.data

_input .word 55555555h *input value*

.text

ldiu r0, st

ldiu *ar2, r1 *load input value*

sti r1, @_input *store input value into _input*

tstb @_input, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_imm/tstb/0

Pattern: patterns/2ops_src_int_imm_src_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)

Random input: 2ops_int_imm/tstb/0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register (value of st)

ldiu r0, st

tstb 0, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_imm/tstb/1

Pattern: patterns/2ops_src_int_imm_src_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)

Random input: 2ops_int_imm/tstb/1/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register (value of st)

ldiu r0, st

tstb 1, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_imm/tstb/-1
Pattern: patterns/2ops_src_int_imm_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)
Random input: 2ops_int_imm/tstb/-1/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 tstb -1, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/tstb/-32768
Pattern: patterns/2ops_src_int_imm_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)
Random input: 2ops_int_imm/tstb/-32768/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 tstb -32768, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_imm/tstb/32767
Pattern: patterns/2ops_src_int_imm_src_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_int_reg.txt (0 tests)
Random input: 2ops_int_imm/tstb/32767/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 tstb 32767, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_reg/addc
Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/addc/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 addc r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_predisp_add
Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addc *++ar2(1)*, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 addc *--ar2(1)*, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 addc *++ar4(1)*, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 addc *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 addc *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 addc *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_postdisp_add_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/addc/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

addc *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/addc/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

addc *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_postdisp_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 addc *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 addc *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 addc *ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
addi 15, ar2 *go at the end of the source buffer*
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu r1, st
addc *-ar2(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the buffer*
ldiu r1, st
addc *++ar4(ir0), r2 *← instruction under test*
ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 addc *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addc *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 addc *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addc *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addc *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addc *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addc *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 addc *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

addi 15, ar2 *go at the end of the source buffer*

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu r1, st

addc *-ar2(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir1 *load ir1 with this random value*

ldiu ar2, ar4 *load a pointer to the buffer*

ldiu r1, st

addc *++ar4(ir1), r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 addc *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addc *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 addc *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addc *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 addc *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 addc *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addc *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addc/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addc *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/addc/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addc/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addc *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/addc
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/addc/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 addc @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/addc/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addc/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addc 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/addc/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addc/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
addc 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/addc/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addc/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
addc -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/addc/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addc/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
addc -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/addc/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/addc/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addc 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/addi

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/addi/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

addi r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/addi/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addi **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

addi *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

addi *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

addi *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 addi *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 addi *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 addi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/addi/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

addi *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/addi/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

addi *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 addi *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 addi *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 addi *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addi *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 addi *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addi *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 addi *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 addi **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 addi *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addi **ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 addi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 addi *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/addi/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addi *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/addi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/addi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addi *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/addi
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/addi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 addi @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/addi/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addi/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addi 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/addi/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addi/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
addi 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/addi/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addi/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
addi -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/addi/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/addi/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
addi -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/addi/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/addi/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addi 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/and

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/and/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

and r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/and/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

and *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

and *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

and *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

and *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 and *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 and *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 and *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/and/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

and *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/and/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

and *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 and *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 and *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 and *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 and *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 and *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 and *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 and *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 and *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 and *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 and *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 and *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 and *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 and *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 and *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 and *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 and *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 and *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 and *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 and *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 and *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 and *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/and/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 and *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/and/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/and/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 and *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/and
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/and/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 and @_input, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/and/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/and/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 and 0, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/and/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/and/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
and 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/and/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/and/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
and -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/and/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/and/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
and -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/and/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/and/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

and 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/andn

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/andn/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

andn r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/andn/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

andn **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

andn *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

andn *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

andn *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 andn *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 andn *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 andn *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/andn/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

andn *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/andn/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

andn *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 andn *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 andn *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 andn *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 andn *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 andn *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 andn *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 andn *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 andn **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 andn *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 andn **ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 andn *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 andn *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 andn *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 andn *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/andn/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 andn **ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/andn/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/andn/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 andn *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/andn
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/andn/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 andn @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/andn/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/andn/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 andn 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/andn/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/andn/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
andn 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/andn/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/andn/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
andn -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/andn/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/andn/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
andn -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/andn/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/andn/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

andn 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/ash

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/ash/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ash r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/ash/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ash **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

ash *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

ash *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

ash *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 ash *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 ash *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ash *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/ash/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

ash *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/ash/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

ash *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 ash *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ash *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 ash *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ash *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ash *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ash *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ash *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ash *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 ash *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ash *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 ash *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ash *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 ash *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 ash *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/ash/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ash *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/ash/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/ash/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 ash *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/ash
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/ash/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 ash @_input, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/ash/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/ash/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 ash 0, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/ash/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/ash/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ash 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/ash/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/ash/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ash -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/ash/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/ash/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ash -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/ash/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/ash/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ash 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/lsh

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/lsh/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

lsh r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/lsh/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

lsh *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

lsh *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

lsh *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

lsh *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 lsh *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 lsh *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 lsh *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/lsh/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

lsh *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/lsh/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

lsh *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 lsh *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 lsh *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 lsh *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lsh *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 lsh *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lsh *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 lsh *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 lsh *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 lsh *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lsh *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 lsh *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lsh *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 lsh *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 lsh *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/lsh/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 lsh *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/lsh/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/lsh/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 lsh *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/lsh
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/lsh/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 lsh @_input, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/lsh/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/lsh/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 lsh 0, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/lsh/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/lsh/1/random.txt (100 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lsh 1, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: 2ops_int_imm/lsh/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/lsh/-1/random.txt (100 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lsh -1, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: 2ops_int_imm/lsh/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/lsh/-32768/random.txt (100 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
lsh -32768, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: 2ops_int_imm/lsh/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/lsh/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

lsh 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/mpyi

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/mpyi/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

mpyi r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/mpyi/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyi **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

mpyi *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

mpyi *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

mpyi *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 mpyi *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 mpyi *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 mpyi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/mpyi/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

mpyi *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/mpyi/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

mpyi *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 mpyi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 mpyi *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 mpyi *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyi *++ar4(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 mpyi *--ar4(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyi *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 mpyi *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 mpyi **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 mpyi *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyi **ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 mpyi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 mpyi *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/mpyi/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 mpyi *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/mpyi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/mpyi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyi *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/mpyi
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/mpyi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 mpyi @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/mpyi/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/mpyi/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 mpyi 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/mpyi/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/mpyi/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
mpyi 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/mpyi/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/mpyi/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
mpyi -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/mpyi/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/mpyi/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
mpyi -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/mpyi/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/mpyi/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyi 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/or

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/or/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

or r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/or/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

or **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

or *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

or *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

or *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 or *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 or *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 or *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/or/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

or *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/or/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

or *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 or *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 or **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 or *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 or *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 or *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 or *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 or *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4--(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 or **+ar2(ir1)*, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 or **-ar2(ir1)*, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 or **+ar4(ir1)*, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 or --ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 or *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 or *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 or *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/or/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 or **ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/or/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/or/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 or *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/or
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/or/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 or @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/or/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/or/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 or 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/or/1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/or/1/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

or 1, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_imm/or/-1

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/or/-1/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

or -1, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_imm/or/-32768

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/or/-32768/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

or -32768, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_imm/or/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/or/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

or 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/subb

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/subb/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subb r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subb/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subb **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

subb *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

subb *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

subb *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subb *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 subb *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subb *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subb/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subb *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subb/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subb *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subb *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subb *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subb *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subb *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subb *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subb *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subb *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subb **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subb *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subb **ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subb *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subb *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subb *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subb *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subb/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subb *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/subb/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subb/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subb *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/subb
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/subb/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 subb @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/subb/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subb/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subb 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/subb/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subb/1/random.txt (100 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subb 1, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: 2ops_int_imm/subb/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subb/-1/random.txt (100 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subb -1, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: 2ops_int_imm/subb/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subb/-32768/random.txt (100 tests)
Assembly pattern under test:
---- *INPUTS* ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- *OUTPUTS* ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subb -32768, r1 ← *instruction under test*
ldiu st, r2

Subdirectory: 2ops_int_imm/subb/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/subb/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subb 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/subc

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/subc/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subc r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subc/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subc **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

subc *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

subc *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

subc *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subc *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 subc *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subc *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subc/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subc *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subc/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subc *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subc *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subc *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subc *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subc *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subc *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subc *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subc *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subc **ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subc *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subc **ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subc *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subc *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subc *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subc *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subc/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subc *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/subc/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subc/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subc *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/subc
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/subc/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 subc @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/subc/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subc/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subc 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/subc/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subc/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subc 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subc/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subc/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subc -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subc/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subc/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subc -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subc/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/subc/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subc 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/subi

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/subi/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subi r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subi/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subi **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

subi *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

subi *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

subi *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subi *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 subi *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subi *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subi/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subi *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subi/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subi *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subi *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subi *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subi *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subi *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subi *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subi *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subi *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subi **ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subi *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subi **ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subi *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subi *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subi *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subi *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subi/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subi *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/subi/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subi/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subi *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/subi
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/subi/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 subi @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/subi/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subi/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subi 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/subi/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subi/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subi 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subi/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subi/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subi -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subi/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subi/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subi -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subi/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/subi/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subi 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/subrb

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/subrb/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subrb r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subrb/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrb *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

subrb *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

subrb *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

subrb *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subrb *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 subrb *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subrb *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_postdisp_sub_circ_mod_bk.4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subrb/indir_postdisp_sub_circ_mod_bk.4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subrb *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_postdisp_add_circ_mod_bk.5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subrb/indir_postdisp_add_circ_mod_bk.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subrb *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subrb *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subrb *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subrb *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrb *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subrb *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrb *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subrb *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subrb *ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subrb *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrb *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subrb *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrb *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subrb *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrb *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subrb/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subrb *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/subrb/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subrb/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrb *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/subrb
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/subrb/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 subrb @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/subrb/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subrb/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subrb 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/subrb/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subrb/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subrb 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subrb/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subrb/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subrb -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subrb/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subrb/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subrb -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subrb/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/subrb/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrb 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/subri

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/subri/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subri r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subri/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subri *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

subri *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

subri *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

subri *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subri *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 subri *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subri *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subri/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subri *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/subri/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

subri *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subri *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subri *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 subri *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subri *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subri *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subri *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subri *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subri *+ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subri *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subri *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subri *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subri *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subri *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subri *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/subri/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subri *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/subri/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/subri/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subri *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/subri
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/subri/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 subri @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/subri/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subri/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subri 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/subri/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subri/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subri 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subri/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subri/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subri -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subri/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/subri/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
subri -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/subri/32767

Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_imm/subri/32767/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subri 32767, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_reg/xor

Pattern: patterns/2ops_src_int_reg_src_dst_int_reg.pat

Manually selected input: inputs/2ops_src_int_reg_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_reg/xor/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

xor r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/xor/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

xor **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_predisp_sub
Pattern: patterns/src_int_indir_predisp_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

addi 16, ar2 *go after the end of the source buffer*

ldiu r0, st

xor *-ar2(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_predisp_add_mod
Pattern: patterns/src_int_indir_predisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

xor *++ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

addi 16, ar4 *go after the end of the source buffer*

ldiu r0, st

xor *--ar4(1), r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 xor *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 xor *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 xor *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_postdisp_sub_circ_mod_bk_4

Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/xor/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

xor *ar4--(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_postdisp_add_circ_mod_bk_5

Pattern: patterns/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.pat

Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_src_dst_int_reg.txt (0 tests)

Random input: 2ops_int_indir/xor/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu 5, bk *load block size (should be at most 8)*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r0, st

xor *ar4++(1)%, r1 *← instruction under test*

ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 xor *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 xor *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 xor *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 xor *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 xor *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 xor *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 xor *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 xor *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 xor *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 xor *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 xor *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 xor *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 xor *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 xor *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_indir/xor/indir
Pattern: patterns/src_int_indir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 xor *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_indir/xor/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_indir/xor/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 xor *ar4++(ir0)b, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_dir/xor
Pattern: patterns/src_int_dir_src_dst_int_reg.pat
Manually selected input: inputs/src_int_dir_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_dir/xor/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h *input value*
 .text
 ldiu r0, st
 ldiu *ar2, r1 *load input value*
 sti r1, @_input *store input value into _input*
 xor @_input, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_int_imm/xor/0
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/xor/0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 xor 0, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_int_imm/xor/1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/xor/1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
xor 1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/xor/-1
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/xor/-1/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
xor -1, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/xor/-32768
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/xor/-32768/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
xor -32768, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_imm/xor/32767
Pattern: patterns/2ops_src_int_imm_src_dst_int_reg.pat
Manually selected input: inputs/2ops_src_int_imm_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_imm/xor/32767/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
xor 32767, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_reg/rol
Pattern: patterns/rot_src_dst_int_reg.pat
Manually selected input: inputs/rot_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/rol/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
rol r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_reg/rolc
Pattern: patterns/rot_src_dst_int_reg.pat
Manually selected input: inputs/rot_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/rolc/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
rolc r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_reg/ror
Pattern: patterns/rot_src_dst_int_reg.pat
Manually selected input: inputs/rot_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/ror/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ror r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_reg/rorc
Pattern: patterns/rot_src_dst_int_reg.pat
Manually selected input: inputs/rot_src_dst_int_reg.txt (0 tests)
Random input: 2ops_int_reg/rorc/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
rorc r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_int_buf/absi_ar_update_ordering
Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 2ops_int_buf/absi_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
absi *ar4++(1), ar4 ← instruction under test
ldiu st, r1

Subdirectory: 2ops_int_buf/negb_ar_update_ordering
Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 2ops_int_buf/negb_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
negb *ar4++(1), ar4 ← instruction under test
ldiu st, r1

Subdirectory: 2ops_int_buf/negi_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/negi_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

negi *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/not_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/not_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

not *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/cmpi_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/cmpi_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

cmpi *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/tstb_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/tstb_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

tstb *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/addc_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/addc_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

addc *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/addi_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/addi_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

addi *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/and_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/and_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

and *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/andn_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/andn_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

andn *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/ash_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/ash_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ash *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/lsh_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/lsh_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

lsh *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/mpyi_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/mpyi_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

mpyi *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/or_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/or_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

or *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/subb_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/subb_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

subb *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/subc_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/subc_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

subc *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/subi_ar_update_ordering

Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 2ops_int_buf/subi_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

subi *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 2ops_int_buf/subrb_ar_update_ordering
Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 2ops_int_buf/subrb_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
subrb *ar4++(1), ar4 ← instruction under test
ldiu st, r1

Subdirectory: 2ops_int_buf/subri_ar_update_ordering
Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 2ops_int_buf/subri_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
subri *ar4++(1), ar4 ← instruction under test
ldiu st, r1

Subdirectory: 2ops_int_buf/xor_ar_update_ordering
Pattern: patterns/2ops_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 2ops_int_buf/xor_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
xor *ar4++(1), ar4 ← instruction under test
ldiu st, r1

Subdirectory: 3ops_int_reg_reg/addi3

Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_reg/addi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register (value of st)

ldiu r0, st

addi3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_int_reg_buf/addi3

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_buf/addi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

addi3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_reg_buf/addi3_ar_update_ordering

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_reg_buf/addi3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

addi3 ar4, *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_reg/addi3

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_reg/addi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

addi3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_buf_reg/addi3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_reg/addi3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

addi3 *ar4++(1), ar4, ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_buf/addi3

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_buf/addi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addi3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_int_buf_buf/addi3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/addi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 addi3 *ar4++(1), *ar4--(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/and3
Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 3ops_int_reg_reg/and3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 and3 r1, r2, r3 ← instruction under test
 ldiu st, r4

Subdirectory: 3ops_int_reg_buf/and3
Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)
Random input: 3ops_int_reg_buf/and3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 and3 r1, *ar2, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_reg_buf/and3_ar_update_ordering
Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_reg_buf/and3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 and3 ar4, *ar4++(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_reg/and3
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 3ops_int_buf_reg/and3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 and3 *ar2, r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_buf_reg/and3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/and3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 and3 *ar4++(1), ar4, ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/and3

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_buf/and3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

and3 *ar2, ++ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_int_buf_buf/and3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_buf/and3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

and3 *ar4++(1), *ar4--(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_reg_reg/andn3

Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_reg/andn3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register (value of st)

ldiu r0, st

andn3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_int_reg_buf/andn3

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_buf/andn3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

andn3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_reg_buf/andn3_ar_update_ordering

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_reg_buf/andn3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

andn3 ar4, *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_reg/andn3

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_reg/andn3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

andn3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_buf_reg/andn3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/andn3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 andn3 *ar4++(1), ar4, ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/andn3
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)
Random input: 3ops_int_buf_buf/andn3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 2 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 andn3 *ar2, **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_int_buf_buf/andn3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/andn3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 andn3 *ar4++(1), *ar4--(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/ash3

Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_reg/ash3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register (value of st)

ldiu r0, st

ash3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_int_reg_buf/ash3

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_buf/ash3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ash3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_reg_buf/ash3_ar_update_ordering

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_reg_buf/ash3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

ash3 ar4, *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_reg/ash3

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_reg/ash3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ash3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_buf_reg/ash3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_reg/ash3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

ash3 *ar4++(1), ar4, ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_buf/ash3

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_buf/ash3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ash3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_int_buf_buf/ash3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/ash3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 ash3 *ar4++(1), *ar4--(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/lsh3
Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 3ops_int_reg_reg/lsh3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 lsh3 r1, r2, r3 ← instruction under test
 ldiu st, r4

Subdirectory: 3ops_int_reg_buf/lsh3
Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)
Random input: 3ops_int_reg_buf/lsh3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 lsh3 r1, *ar2, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_reg_buf/lsh3_ar_update_ordering
Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_reg_buf/lsh3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 lsh3 ar4, *ar4++(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_reg/lsh3
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 3ops_int_buf_reg/lsh3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 lsh3 *ar2, r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_buf_reg/lsh3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/lsh3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 lsh3 *ar4++(1), ar4, ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/lsh3

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_buf/lsh3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

lsh3 *ar2, ++ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_int_buf_buf/lsh3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_buf/lsh3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

lsh3 *ar4++(1), *ar4--(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_reg_reg/mpyi3

Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_reg/mpyi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register (value of st)

ldiu r0, st

mpyi3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_int_reg_buf/mpyi3

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_buf/mpyi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

mpyi3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_reg_buf/mpyi3_ar_update_ordering

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_reg_buf/mpyi3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

mpyi3 ar4, *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_reg/mpyi3

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_reg/mpyi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

mpyi3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_buf_reg/mpyi3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/mpyi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 mpyi3 *ar4++(1), ar4, ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/mpyi3
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)
Random input: 3ops_int_buf_buf/mpyi3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 2 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 mpyi3 *ar2, **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_int_buf_buf/mpyi3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/mpyi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 mpyi3 *ar4++(1), *ar4--(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/or3

Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_reg/or3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register (value of st)

ldiu r0, st

or3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_int_reg_buf/or3

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_buf/or3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

or3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_reg_buf/or3_ar_update_ordering

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_reg_buf/or3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

or3 ar4, *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_reg/or3

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_reg/or3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

or3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_buf_reg/or3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_reg/or3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

or3 *ar4++(1), ar4, ar2 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_buf/or3

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_buf/or3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

or3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_int_buf_buf/or3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/or3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 or3 *ar4++(1), *ar4--(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/subb3
Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 3ops_int_reg_reg/subb3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 subb3 r1, r2, r3 ← instruction under test
 ldiu st, r4

Subdirectory: 3ops_int_reg_buf/subb3
Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)
Random input: 3ops_int_reg_buf/subb3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 subb3 r1, *ar2, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_reg_buf/subb3_ar_update_ordering
Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_reg_buf/subb3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 subb3 ar4, *ar4++(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_reg/subb3
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)
Random input: 3ops_int_buf_reg/subb3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 subb3 *ar2, r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_buf_reg/subb3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/subb3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 subb3 *ar4++(1), ar4, ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/subb3

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_buf/subb3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subb3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_int_buf_buf/subb3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_buf/subb3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

subb3 *ar4++(1), *ar4--(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_reg_reg/subi3

Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_reg/subi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register (value of st)

ldiu r0, st

subi3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_int_reg_buf/subi3

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_buf/subi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subi3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_reg_buf/subi3_ar_update_ordering

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_reg_buf/subi3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

subi3 ar4, *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_reg/subi3

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_reg/subi3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subi3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_buf_reg/subi3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/subi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 subi3 *ar4++(1), ar4, ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/subi3
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)
Random input: 3ops_int_buf_buf/subi3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 2 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subi3 *ar2, **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_int_buf_buf/subi3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/subi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 subi3 *ar4++(1), *ar4--(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/xor3

Pattern: patterns/3ops_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_reg/xor3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r2: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register (value of st)

ldiu r0, st

xor3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_int_reg_buf/xor3

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_reg_buf/xor3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

xor3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_reg_buf/xor3_ar_update_ordering

Pattern: patterns/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_reg_buf/xor3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

xor3 ar4, *ar4++(1), ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_reg/xor3

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_reg/xor3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r1: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

xor3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_int_buf_reg/xor3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_reg/xor3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

xor3 *ar4++(1), ar4, ar4 ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_buf/xor3

Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)

Random input: 3ops_int_buf_buf/xor3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

xor3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_int_buf_buf/xor3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/xor3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 xor3 *ar4++(1), *ar4--(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/cmpi3
Pattern: patterns/3ops_src_int_reg_src_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_reg.txt (0 tests)
Random input: 3ops_int_reg_reg/cmpi3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpi3 r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_reg_buf/cmpi3
Pattern: patterns/3ops_src_int_reg_src_int_buf.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf.txt (0 tests)
Random input: 3ops_int_reg_buf/cmpi3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpi3 r1, *ar2 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_int_reg_buf/cmpi3_ar_update_ordering
Pattern: patterns/3ops_src_int_reg_src_int_buf_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_reg_buf/cmpi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 cmpi3 ar4, *ar4++(1) ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_reg/cmpi3
Pattern: patterns/3ops_src_int_buf_src_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg.txt (0 tests)
Random input: 3ops_int_buf_reg/cmpi3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpi3 *ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_int_buf_reg/cmpi3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/cmpi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 cmpi3 *ar4++(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/cmpi3
Pattern: patterns/3ops_src_int_buf_src_int_buf.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf.txt (0 tests)
Random input: 3ops_int_buf_buf/cmpi3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 2 32-bit integer values
 ---- OUTPUTS ----
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpi3 *ar2, **ar2(1) ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/cmpi3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_buf_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_buf_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_buf/cmpi3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 cmpi3 *ar4++(1), *ar4--(1) ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_reg_reg/tstb3
Pattern: patterns/3ops_src_int_reg_src_int_reg.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_reg.txt (0 tests)
Random input: 3ops_int_reg_reg/tstb3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 tstb3 r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_int_reg_buf/tstb3
Pattern: patterns/3ops_src_int_reg_src_int_buf.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf.txt (0 tests)
Random input: 3ops_int_reg_buf/tstb3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 tstb3 r1, *ar2 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_int_reg_buf/tstb3_ar_update_ordering
Pattern: patterns/3ops_src_int_reg_src_int_buf_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_reg_src_int_buf_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_reg_buf/tstb3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 tstb3 ar4, *ar4++(1) ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_reg/tstb3
Pattern: patterns/3ops_src_int_buf_src_int_reg.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg.txt (0 tests)
Random input: 3ops_int_buf_reg/tstb3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 tstb3 *ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_int_buf_reg/tstb3_ar_update_ordering
Pattern: patterns/3ops_src_int_buf_src_int_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_int_buf_src_int_reg_ar_update_ordering.txt (0 tests)
Random input: 3ops_int_buf_reg/tstb3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 tstb3 *ar4++(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_int_buf_buf/tstb3

Pattern: patterns/3ops_src_int_buf_src_int_buf.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf.txt (0 tests)

Random input: 3ops_int_buf_buf/tstb3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register (value of st)

ldiu r0, st

tstb3 *ar2, **ar2(1) ← instruction under test

ldiu st, r1

Subdirectory: 3ops_int_buf_buf/tstb3_ar_update_ordering

Pattern: patterns/3ops_src_int_buf_src_int_buf_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_int_buf_src_int_buf_ar_update_ordering.txt (0 tests)

Random input: 3ops_int_buf_buf/tstb3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

tstb3 *ar4++(1), *ar4--(1) ← instruction under test

ldiu st, r1

Subdirectory: parallel_int/ldi_ldi/buf_buf

Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_buf_dst_int_reg.txt (0 tests)

Random input: parallel_int/ldi_ldi/buf_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

ar4: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldi *ar2, r1

|| ldi *ar4, r2

ldiu st, r3

Subdirectory: parallel_int/ldi_ldi_ar_update_ordering/buf_buf
Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/ldi_ldi_ar_update_ordering/buf_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 ldi *ar4++(1), r1
 || ldi *ar4--(1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/ldi_ldi/reg_buf
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_buf_dst_int_reg.txt (0 tests)
Random input: parallel_int/ldi_ldi/reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 ldi r1, r2
 || ldi *ar2, r3
 ldiu st, r4

Subdirectory: parallel_int/ldi_ldi_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/ldi_ldi_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 ldi ar4, r1
 || ldi *ar4++(1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/ldi_sti/buf_buf

Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat

Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)

Random input: parallel_int/ldi_sti/buf_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

r1: a 32-bit integer register

ar4: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldi *ar2, r1

|| sti r2, *ar4 ← instruction under test

ldiu st, r3

Subdirectory: parallel_int/ldi_sti_ar_update_ordering/buf_buf

Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat

Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)

Random input: parallel_int/ldi_sti_ar_update_ordering/buf_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar2, ar5

ldi *ar4++(1), r1

|| sti r2, *ar4--(1) ← instruction under test

ldiu st, r3

Subdirectory: parallel_int/ldi_sti/reg_buf

Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat

Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)

Random input: parallel_int/ldi_sti/reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 32-bit integer register

r3: a 32-bit integer register

---- OUTPUTS ----

r2: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r4: a 32-bit integer register (value of st)

ldiu r0, st

ldi r1, r2

|| sti r3, *ar2 ← instruction under test

ldiu st, r4

Subdirectory: parallel_int/ldi_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/ldi_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 ldi ar4, r1
 || sti r2, *ar4++(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/sti_sti/buf_buf
Pattern: patterns/par_src_int_reg_dst_int_buf_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_dst_int_buf_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/sti_sti/buf_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 sti r1, *ar2
 || sti r2, *ar4 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/sti_sti_ar_update_ordering/buf_buf
Pattern: patterns/par_src_int_reg_dst_int_buf_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_reg_dst_int_buf_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/sti_sti_ar_update_ordering/buf_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 sti r1, *ar4++(1)
 || sti r2, *ar4--(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/sti_sti/reg_buf
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/sti_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r3: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 sti r1, r2
 || sti r3, *ar2 ← instruction under test
 ldiu st, r4

Subdirectory: parallel_int/sti_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/sti_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 sti ar4, r1
 || sti r2, *ar4++(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/absi_sti/buf_reg
Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/absi_sti/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 absi *ar2, r1
 || sti r2, *ar4 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/absi_sti_ar_update_ordering/buf_reg
Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/absi_sti_ar_update_ordering/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 absi *ar4++(1), r1
 || sti r2, *ar4--(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/absi_sti/reg_reg
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/absi_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r3: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 absi r1, r2
 || sti r3, *ar2 ← instruction under test
 ldiu st, r4

Subdirectory: parallel_int/absi_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/absi_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 absi ar4, r1
 || sti r2, *ar4++(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/negi_sti/buf_reg
Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/negi_sti/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 negi *ar2, r1
 || sti r2, *ar4 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/negi_sti_ar_update_ordering/buf_reg
Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/negi_sti_ar_update_ordering/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 negi *ar4++(1), r1
 || sti r2, *ar4--(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/negi_sti/reg_reg
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/negi_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r3: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 negi r1, r2
 || sti r3, *ar2 ← instruction under test
 ldiu st, r4

Subdirectory: parallel_int/negi_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/negi_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 negi ar4, r1
 || sti r2, *ar4++(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/not_sti/buf_reg
Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/not_sti/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
r1: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 not *ar2, r1
 || sti r2, *ar4 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/not_sti_ar_update_ordering/buf_reg
Pattern: patterns/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/not_sti_ar_update_ordering/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 not *ar4++(1), r1
 || sti r2, *ar4--(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/not_sti/reg_reg
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/not_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r3: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 not r1, r2
 || sti r3, *ar2 ← instruction under test
 ldiu st, r4

Subdirectory: parallel_int/not_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.pat
Manually selected input: inputs/par_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.txt (0 tests)
Random input: parallel_int/not_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 not ar4, r1
 || sti r2, *ar4++(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_int/addi3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.txt (0 tests)
Random input: parallel_int/addi3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
 ldiu r0, st
 addi3 r1, *ar2, r2
 || sti r3, *ar4 ← instruction under test
 ldiu st, r4

Subdirectory: parallel_int/addi3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/addi3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
addi3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/addi3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/addi3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
addi3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/addi3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/addi3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
addi3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/and3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/and3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
ldiu r0, st
and3 r1, *ar2, r2
|| sti r3, *ar4 ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/and3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/and3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
and3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/and3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/and3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
and3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/and3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/and3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
and3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/ash3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/ash3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
ldiu r0, st
ash3 r1, *ar2, r2
|| sti r3, *ar4 ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/ash3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/ash3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
ash3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/ash3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/ash3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
ash3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/ash3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/ash3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
ash3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/lsh3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/lsh3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
ldiu r0, st
lsh3 r1, *ar2, r2
|| sti r3, *ar4 ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/lsh3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/lsh3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
lsh3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/lsh3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/lsh3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
lsh3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/lsh3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/lsh3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
lsh3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/mpyi3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/mpyi3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
ldiu r0, st
mpyi3 r1, *ar2, r2
|| sti r3, *ar4 ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/mpyi3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
mpyi3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/mpyi3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/mpyi3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
mpyi3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/mpyi3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
mpyi3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/or3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/or3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
ldiu r0, st
or3 r1, *ar2, r2
|| sti r3, *ar4 ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/or3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/or3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
or3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/or3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/or3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
or3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/or3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/or3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
or3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/subi3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/subi3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
ldiu r0, st
subi3 r1, *ar2, r2
|| sti r3, *ar4 ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/subi3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/subi3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
subi3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/subi3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/subi3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
subi3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/subi3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/subi3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
subi3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/xor3_sti/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/xor3_sti/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
---- OUTPUTS ----
r2: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register (value of st)
ldiu r0, st
xor3 r1, *ar2, r2
|| sti r3, *ar4 ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/xor3_sti_ar_update_ordering/reg_buf
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/xor3_sti_ar_update_ordering/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
xor3 r1, *ar4++(1), r2
|| sti ar4, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_int/xor3_sti/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf.
↪ txt (0 tests)
Random input: parallel_int/xor3_sti/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r2: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register (value of st)
ldiu r0, st
xor3 r1, r2, r3
|| sti r4, *ar2 ← instruction under test
ldiu st, r5

Subdirectory: parallel_int/xor3_sti_ar_update_ordering/reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_ar_update_ordering.
↪ pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_buf_
↪ ar_update_ordering.txt (0 tests)
Random input: parallel_int/xor3_sti_ar_update_ordering/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r1: a 32-bit integer register
r2: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
xor3 r1, r2, r3
|| sti ar4, *ar4++(1) ← instruction under test
ldiu st, r4

Subdirectory: parallel_int/mpyi3_addi3/buf_buf_reg_reg
Pattern: patterns/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_
↪ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_addi3/buf_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register (value of st)
ldiu r1, st
mpyi3 *ar2, *ar4, r0
|| addi3 r3, r4, r2 ← instruction under test
ldiu r0, r3
ldiu r2, r4
ldiu st, r5

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/buf_buf_reg_reg
Pattern: patterns/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg_ar_↵
↵ update_ordering.pat
Manually selected input: inputs/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_↵
↵ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_addi3_ar_update_ordering/buf_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
ldiu r1, st
ldiu ar2, ar4
ldiu ar2, ar5
mpyi3 *ar4++(1), *ar4--(1), r0
|| addi3 r3, r4, r2 ← instruction under test
ldiu r0, r5
ldiu r2, r6
ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/buf_reg_reg_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_↵
↵ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_addi3/buf_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register
---- OUTPUTS ----
r4: a 32-bit integer register
r5: a 32-bit integer register
r6: a 32-bit integer register (value of st)
ldiu r1, st
mpyi3 *ar2, r3, r0
|| addi3 r4, r5, r2 ← instruction under test
ldiu r0, r4
ldiu r2, r5
ldiu st, r6

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/buf_reg_reg_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg_ar_↪ update_ordering.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_↪ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_addi3_ar_update_ordering/buf_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 *ar4++(1), r3, r0
 || addi3 ar4, r4, r2 ↪ instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/reg_buf_reg_reg
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_↪ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_addi3/reg_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register
r5: a 32-bit integer register
 ---- OUTPUTS ----
r4: a 32-bit integer register
r5: a 32-bit integer register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 r3, *ar2, r0
 || addi3 r4, r5, r2 ↪ instruction under test
 ldiu r0, r4
 ldiu r2, r5
 ldiu st, r6

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/reg.buf.reg.reg
Pattern: patterns/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg_ar_↪ update_ordering.pat
Manually selected input: inputs/par_src_int_reg_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_↪ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_addi3_ar_update_ordering/reg.buf.reg.reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 ar4, *ar4++(1), r0
 || addi3 r3, r4, r2 ↪ instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/reg.reg.reg.reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_↪ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_addi3/reg.reg.reg.reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register
r6: a 32-bit integer register
 ---- OUTPUTS ----
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 r3, r4, r0
 || addi3 r5, r6, r2 ↪ instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/buf_reg_buf_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_↵
 ↵ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_addi3/buf_reg_buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register
 ---- OUTPUTS ----
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 *ar2, r3, r0
 || addi3 *ar4, r4, r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r4
 ldiu st, r5

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/buf_reg_buf_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg_ar_↵
 ↵ update_ordering.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_↵
 ↵ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_addi3_ar_update_ordering/buf_reg_buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 *ar4++(1), r3, r0
 || addi3 *ar4--(1), r4, r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/reg_reg_buf_reg

Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_↵ dst_int_reg.txt (0 tests)

Random input: parallel_int/mpyi3_addi3/reg_reg_buf_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

r3: a 32-bit integer register

r4: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r5: a 32-bit integer register

---- OUTPUTS ----

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register (value of st)

ldiu r1, st

mpyi3 r3, r4, r0

|| addi3 *ar2, r5, r2 ← instruction under test

ldiu r0, r4

ldiu r2, r5

ldiu st, r6

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/reg_reg_buf_reg

Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg_ar_↵ update_ordering.pat

Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_↵ dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: parallel_int/mpyi3_addi3_ar_update_ordering/reg_reg_buf_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register

r4: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register (value of st)

ldiu r1, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyi3 ar4, r3, r0

|| addi3 *ar4++(1), r4, r2 ← instruction under test

ldiu r0, r5

ldiu r2, r6

ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/reg_reg_buf_buf

Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_↵
↵ dst_int_reg.txt (0 tests)

Random input: parallel_int/mpyi3_addi3/reg_reg_buf_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

r3: a 32-bit integer register

r4: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

ar4: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register (value of st)

ldiu r1, st

mpyi3 r3, r4, r0

|| addi3 *ar2, *ar4, r2 ← instruction under test

ldiu r0, r3

ldiu r2, r4

ldiu st, r5

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/reg_reg_buf_buf

Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_dst_int_reg_ar_↵
↵ update_ordering.pat

Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_↵
↵ dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: parallel_int/mpyi3_addi3_ar_update_ordering/reg_reg_buf_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register

r4: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register (value of st)

ldiu r1, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyi3 r3, r4, r0

|| addi3 *ar4++(1), *ar4--(1), r2 ← instruction under test

ldiu r0, r5

ldiu r2, r6

ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/reg_reg_reg_buf

Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_↵
↵ dst_int_reg.txt (0 tests)

Random input: parallel_int/mpyi3_addi3/reg_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r4: a 32-bit integer register

r5: a 32-bit integer register

r6: a 32-bit integer register (value of st)

ldiu r1, st

mpyi3 r3, r4, r0

|| addi3 r5, *ar2, r2 ↵ *instruction under test*

ldiu r0, r3

ldiu r2, r5

ldiu st, r6

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/reg_reg_reg_buf

Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg_ar_↵
↵ update_ordering.pat

Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_↵
↵ dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: parallel_int/mpyi3_addi3_ar_update_ordering/reg_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register

r4: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register (value of st)

ldiu r1, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyi3 ar4, r3, r0

|| addi3 r4, *ar4++(1), r2 ↵ *instruction under test*

ldiu r0, r5

ldiu r2, r6

ldiu st, r7

Subdirectory: parallel_int/mpyi3_addi3/buf_reg_reg_buf

Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg.pat

Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_↵
↵ dst_int_reg.txt (0 tests)

Random input: parallel_int/mpyi3_addi3/buf_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register

r4: a 32-bit integer register

ar4: an auxiliary register pointing to an array of 1 32-bit integer values

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register (value of st)

ldiu r1, st

mpyi3 *ar2, r3, r0

|| addi3 r4, *ar4, r2 ← instruction under test

ldiu r0, r3

ldiu r2, r4

ldiu st, r5

Subdirectory: parallel_int/mpyi3_addi3_ar_update_ordering/buf_reg_reg_buf

Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg_ar_↵
↵ update_ordering.pat

Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_↵
↵ dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: parallel_int/mpyi3_addi3_ar_update_ordering/buf_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register

r4: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register (value of st)

ldiu r1, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyi3 *ar4++(1), r3, r0

|| addi3 r4, *ar4--(1), r2 ← instruction under test

ldiu r0, r5

ldiu r2, r6

ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/buf_buf_reg_reg

Pattern: patterns/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg.pat

Manually selected input: inputs/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_↵ dst_int_reg.txt (0 tests)

Random input: parallel_int/mpyi3_subi3/buf_buf_reg_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

ar4: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register

r4: a 32-bit integer register

---- OUTPUTS ----

r3: a 32-bit integer register

r4: a 32-bit integer register

r5: a 32-bit integer register (value of st)

ldiu r1, st

mpyi3 *ar2, *ar4, r0

|| subi3 r3, r4, r2 ← instruction under test

ldiu r0, r3

ldiu r2, r4

ldiu st, r5

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/buf_buf_reg_reg

Pattern: patterns/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg_ar_↵ update_ordering.pat

Manually selected input: inputs/par_src_int_buf_src_int_buf_dst_int_reg_src_int_reg_src_int_reg_↵ dst_int_reg_ar_update_ordering.txt (0 tests)

Random input: parallel_int/mpyi3_subi3_ar_update_ordering/buf_buf_reg_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register

r4: a 32-bit integer register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r5: a 32-bit integer register

r6: a 32-bit integer register

r7: a 32-bit integer register (value of st)

ldiu r1, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyi3 *ar4++(1), *ar4--(1), r0

|| subi3 r3, r4, r2 ← instruction under test

ldiu r0, r5

ldiu r2, r6

ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/buf_reg_reg_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_↵
 ↵ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/buf_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register
 ---- OUTPUTS ----
r4: a 32-bit integer register
r5: a 32-bit integer register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 *ar2, r3, r0
 || subi3 r4, r5, r2 ↵ instruction under test
 ldiu r0, r4
 ldiu r2, r5
 ldiu st, r6

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/buf_reg_reg_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg_ar_↵
 ↵ update_ordering.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_↵
 ↵ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_subi3_ar_update_ordering/buf_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 *ar4++(1), r3, r0
 || subi3 ar4, r4, r2 ↵ instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/reg.buf.reg.reg
Pattern: patterns/par_src_int_reg_src_int.buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_src_int.buf_dst_int_reg_src_int_reg_src_int_reg_↵
 ↵ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/reg.buf.reg.reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r4: a 32-bit integer register
r5: a 32-bit integer register
 ---- OUTPUTS ----
r4: a 32-bit integer register
r5: a 32-bit integer register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 r3, *ar2, r0
 || subi3 r4, r5, r2 ← instruction under test
 ldiu r0, r4
 ldiu r2, r5
 ldiu st, r6

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/reg.buf.reg.reg
Pattern: patterns/par_src_int_reg_src_int.buf_dst_int_reg_src_int_reg_src_int_reg_dst_int_reg_ar_↵
 ↵ update_ordering.pat
Manually selected input: inputs/par_src_int_reg_src_int.buf_dst_int_reg_src_int_reg_src_int_reg_↵
 ↵ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_subi3_ar_update_ordering/reg.buf.reg.reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 ar4, *ar4++(1), r0
 || subi3 r3, r4, r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/reg_reg_reg_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_reg_↵
 ↵ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/reg_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r1: a 32-bit integer register (value for st)
 r3: a 32-bit integer register
 r4: a 32-bit integer register
 r5: a 32-bit integer register
 r6: a 32-bit integer register
 ---- OUTPUTS ----
 r5: a 32-bit integer register
 r6: a 32-bit integer register
 r7: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 r3, r4, r0
 || subi3 r5, r6, r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/buf_reg_buf_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_↵
 ↵ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/buf_reg_buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r1: a 32-bit integer register (value for st)
 ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 r3: a 32-bit integer register
 ar4: an auxiliary register pointing to an array of 1 32-bit integer values
 r4: a 32-bit integer register
 ---- OUTPUTS ----
 r3: a 32-bit integer register
 r4: a 32-bit integer register
 r5: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 *ar2, r3, r0
 || subi3 *ar4, r4, r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r4
 ldiu st, r5

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/buf_reg_buf_reg
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg_ar_↪ update_ordering.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_↪ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_subi3_ar_update_ordering/buf_reg_buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 *ar4++(1), r3, r0
 || subi3 *ar4--(1), r4, r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/reg_reg_buf_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_↪ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/reg_reg_buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 32-bit integer register
r4: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r5: a 32-bit integer register
 ---- OUTPUTS ----
r4: a 32-bit integer register
r5: a 32-bit integer register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 r3, r4, r0
 || subi3 *ar2, r5, r2 ← instruction under test
 ldiu r0, r4
 ldiu r2, r5
 ldiu st, r6

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/reg_reg_buf_reg
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg_ar_ ↪ update_ordering.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_reg_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_subi3_ar_update_ordering/reg_reg_buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 ar4, r3, r0
 || subi3 *ar4++(1), r4, r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/reg_reg_buf_buf
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/reg_reg_buf_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 32-bit integer register
r4: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 r3, r4, r0
 || subi3 *ar2, *ar4, r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r4
 ldiu st, r5

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/reg_reg_buf_buf
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_dst_int_reg_ar_ ↪ update_ordering.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_buf_src_int_buf_ ↪ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_subi3_ar_update_ordering/reg_reg_buf_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 r3, r4, r0
 || subi3 *ar4++(1), *ar4--(1), r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/reg_reg_reg_buf
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_ ↪ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/reg_reg_reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r4: a 32-bit integer register
r5: a 32-bit integer register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 r3, r4, r0
 || subi3 r5, *ar2, r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r5
 ldiu st, r6

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/reg_reg_reg_buf
Pattern: patterns/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg_ar_ ↪ update_ordering.pat
Manually selected input: inputs/par_src_int_reg_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_ ↪ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_subi3_ar_update_ordering/reg_reg_reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 ar4, r3, r0
 || subi3 r4, *ar4++(1), r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: parallel_int/mpyi3_subi3/buf_reg_reg_buf
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_ ↪ dst_int_reg.txt (0 tests)
Random input: parallel_int/mpyi3_subi3/buf_reg_reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
ar4: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r3: a 32-bit integer register
r4: a 32-bit integer register
r5: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyi3 *ar2, r3, r0
 || subi3 r4, *ar4, r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r4
 ldiu st, r5

Subdirectory: parallel_int/mpyi3_subi3_ar_update_ordering/buf_reg_reg_buf
Pattern: patterns/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_dst_int_reg_ar_ ↪ update_ordering.pat
Manually selected input: inputs/par_src_int_buf_src_int_reg_dst_int_reg_src_int_reg_src_int_buf_ ↪ dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: parallel_int/mpyi3_subi3_ar_update_ordering/buf_reg_reg_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r3: a 32-bit integer register
r4: a 32-bit integer register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r5: a 32-bit integer register
r6: a 32-bit integer register
r7: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyi3 *ar4++(1), r3, r0
 || subi3 r4, *ar4--(1), r2 ← instruction under test
 ldiu r0, r5
 ldiu r2, r6
 ldiu st, r7

Subdirectory: 2ops_float_indir/absf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absf *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 absf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 absf *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 go after the end of the source buffer
 ldiu r0, st
 absf *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 absf *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 absf *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 absf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 absf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 absf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 absf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 absf *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 absf *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 absf *++ar4(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the source buffer
 addi 16, ar4 go just after the end of the source buffer
 ldiu r1, st
 absf *--ar4(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 absf *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 absf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 absf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 absf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 absf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 absf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 absf **ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 absf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 absf **ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to the source buffer
 addi 16, ar4 go just after the end of the source buffer
 ldiu r1, st
 absf *--ar4(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 absf *ar4++(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to the source buffer
 addi 15, ar4 go at the end of the source buffer
 ldiu r1, st
 absf *ar4--(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absf *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/absf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 absf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/absf/indir

Pattern: patterns/src_float_indir_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/absf/indir/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

absf **ar2, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_indir/absf/indir_postind_ir0_add_bit_rev_mod

Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/absf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

and 15, r0 crop random value between 0 and 15

ldiu r0, ir0 load ir0 with this random value

ldiu ar2, ar4 load a pointer to the buffer

ldiu r1, st

absf *ar4++(ir0)b, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_float_reg/absf

Pattern: patterns/2ops_src_float_reg_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_reg_dst_float_reg.txt (0 tests)

Random input: 2ops_float_reg/absf/random.txt (10000 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu r0, st

absf r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 2ops_float_imm/absf/0.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/0.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absf 0.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/absf/1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absf 1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/absf/-1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/-1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absf -1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/absf/1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absf 1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/absf/-1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
absf -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/absf/2.5594e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
absf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/absf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
absf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/absf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
absf -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/absf/-2.56e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/absf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 absf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/absf
Pattern: patterns/src_float_dir_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/absf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 absf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 *go after the end of the source buffer*
 ldiu r0, st
 negf *-ar2(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 negf ++ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 negf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 negf *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 go at the end of the source buffer
 ldiu r0, st
 negf *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 negf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 negf **ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 negf *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negf **ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 negf *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negf *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 negf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 negf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 negf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 negf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 negf *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 negf *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negf *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 negf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 negf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 negf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 negf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 negf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 negf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 negf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/negf/indir
Pattern: patterns/src_float_indir_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/negf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/negf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 negf *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_reg/negf
Pattern: patterns/2ops_src_float_reg_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/negf/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 negf r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_imm/negf/0.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/0.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf 0.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf 1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/-1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/-1.0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf -1.0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf 1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/-1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/-1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf -1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/2.5594e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf 2.5594e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf 7.8125e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/negf/-2.56e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/negf/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 negf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/negf
Pattern: patterns/src_float_dir_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/negf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 negf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 norm *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 norm *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 norm *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 norm *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 norm *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 norm *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 norm *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 norm *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 norm *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 norm *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 norm *ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 norm *ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 norm *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 norm *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 norm *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 norm *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 norm *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 norm *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 norm *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 norm *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 norm *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 norm *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 norm *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 norm *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 norm *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 norm *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 norm *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 norm *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 norm *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 norm *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/norm/indir
Pattern: patterns/src_float_indir_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 norm *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/norm/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/norm/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 norm *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_reg/norm
Pattern: patterns/2ops_src_float_reg_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/norm/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
norm r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_imm/norm/0.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
norm 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/norm/1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
norm 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/norm/-1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
norm -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/norm/1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
norm 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/norm/-1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
norm -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/norm/2.5594e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
norm 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/norm/7.8125e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
norm 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/norm/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 norm -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/norm/-2.56e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/norm/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 norm -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/norm
Pattern: patterns/src_float_dir_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/norm/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 norm @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 rnd *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 rnd *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 rnd *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 rnd *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 rnd *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 rnd *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 rnd *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 rnd *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 rnd *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 rnd *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 rnd *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 rnd *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 rnd *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 rnd *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 rnd *ar4--(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 rnd *ar4++(ir0)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 rnd *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 rnd *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 rnd *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 rnd *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 rnd *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 rnd *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 rnd *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 rnd *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 rnd *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 rnd *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 rnd *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 rnd *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 rnd *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/rnd/indir
Pattern: patterns/src_float_indir_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/rnd/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/rnd/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 rnd *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_reg/rnd
Pattern: patterns/2ops_src_float_reg_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/rnd/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
rnd r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_imm/rnd/0.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
rnd 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
rnd 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/-1.0
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
rnd -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd 1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/-1.5
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/-1.5/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd -1.5, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/2.5594e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd 2.5594e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/7.8125e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd 7.8125e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/rnd/-2.56e2
Pattern: patterns/2ops_src_float_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/rnd/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 rnd -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/rnd
Pattern: patterns/src_float_dir_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/rnd/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
 r2: a 40-bit floating point register
 r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 rnd @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 addf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 addf **ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 addf *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 addf *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 addf *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 addf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 addf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 addf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 addf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 addf **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 addf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addf **ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 addf *--ar4(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addf *ar4++(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 addf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 addf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 addf *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_preind_ir1_sub

Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/addf/indir_preind_ir1_sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register

r1: a 32-bit integer register (value for st)

r2: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

addi 15, ar2 go at the end of the source buffer

and 15, r0 crop random value between 0 and 15

ldiu r0, ir1 load ir1 with this random value

ldiu r1, st

*addf *-ar2(ir1), r2 ← instruction under test*

ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_preind_ir1_add_mod

Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat

Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/addf/indir_preind_ir1_add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r1: a 32-bit integer register (value for st)

r2: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

and 15, r0 crop random value between 0 and 15

ldiu r0, ir1 load ir1 with this random value

ldiu ar2, ar4 load a pointer to the buffer

ldiu r1, st

*addf *++ar4(ir1), r2 ← instruction under test*

ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 addf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 addf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 addf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 addf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 addf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 addf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 addf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/addf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/addf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addf *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/addf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: 2ops_float_indir/addf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
addf *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_reg/addf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/addf/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
addf r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_imm/addf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/addf/0.0/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
addf 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/addf/1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/addf/1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addf 1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/addf/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/addf/-1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addf -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/addf/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/addf/1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addf 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/addf/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/addf/-1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addf -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/addf/2.5594e2

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/addf/2.5594e2/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addf 2.5594e2, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/addf/7.8125e-3

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/addf/7.8125e-3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addf 7.8125e-3, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/addf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/addf/-7.8163e-3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 40-bit floating point register
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addf -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/addf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/addf/-2.56e2/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 40-bit floating point register
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 addf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/addf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/addf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 r2: a 40-bit floating point register
 ---- OUTPUTS ----
 r2: a 40-bit floating point register
 r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 addf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 subf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subf **ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 subf *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subf *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 subf *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 subf *ar4++(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r0, st
 subf *ar4--(1)%, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 subf **ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 subf *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subf **ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subf *--ar4(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subf *ar4++(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 subf *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subf **ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subf *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_reg/subf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/subf/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 subf r1, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_imm/subf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/subf/0.0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subf 0.0, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_imm/subf/1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subf/1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subf 1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subf/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subf/-1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subf -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subf/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subf/1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subf 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subf/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subf/-1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subf -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subf/2.5594e2

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subf/2.5594e2/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subf 2.5594e2, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subf/7.8125e-3

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subf/7.8125e-3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subf 7.8125e-3, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/subf/-7.8163e-3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subf -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/subf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/subf/-2.56e2/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/subf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/subf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 subf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subrf *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 subrf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subrf *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 subrf *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 subrf *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 subrf *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subrf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subrf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subrf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 subrf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir0.add
Pattern: patterns/src_float_indir_preind_ir0.add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir0.add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 subrf *+ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir0.sub
Pattern: patterns/src_float_indir_preind_ir0.sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir0.sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 subrf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir0.add_mod
Pattern: patterns/src_float_indir_preind_ir0.add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir0.add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrf *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subrf *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrf *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subrf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir0_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir0_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subrf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir0_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir0_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subrf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir0_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir0_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrf *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subrf *+ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 subrf *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrf *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 subrf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 subrf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir1_sub_circ_mod_bk.4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir1_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subrf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir1_add_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir1_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 subrf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir1_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir1_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 subrf *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/subrf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subrf *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/subrf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/subrf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 subrf *ar4++(ir0)b, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_reg/subrf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/subrf/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 subrf r1, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_imm/subrf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/subrf/0.0/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subrf 0.0, r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subrf/1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrf 1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subrf/-1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrf -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subrf/1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrf 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subrf/-1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrf -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/2.5594e2

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subrf/2.5594e2/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrf 2.5594e2, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/7.8125e-3

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/subrf/7.8125e-3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subrf 7.8125e-3, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/subrf/-7.8163e-3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subrf -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/subrf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/subrf/-2.56e2/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 subrf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/subrf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/subrf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ld fu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 subrf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 mpyf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 mpyf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 mpyf **ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 mpyf *--ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 mpyf *ar4++(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 mpyf *ar4--(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 mpyf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 mpyf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 mpyf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 mpyf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir0.add
Pattern: patterns/src_float_indir_preind_ir0.add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir0.add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 mpyf **ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir0.sub
Pattern: patterns/src_float_indir_preind_ir0.sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir0.sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 mpyf *-ar2(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir0.add_mod
Pattern: patterns/src_float_indir_preind_ir0.add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0.add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir0.add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyf **ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 mpyf *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyf *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 mpyf *ar4--(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyf *ar4++(ir0)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 mpyf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 mpyf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 mpyf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 mpyf *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 mpyf *-ar2(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyf *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 mpyf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 mpyf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 mpyf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 mpyf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 mpyf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 mpyf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 mpyf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/mpyf/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 mpyf *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/mpyf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: 2ops_float_indir/mpyf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
mpyf *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_reg/mpyf
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/mpyf/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
mpyf r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_imm/mpyf/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/mpyf/0.0/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
mpyf 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/mpyf/1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyf 1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/mpyf/-1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyf -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/mpyf/1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyf 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/mpyf/-1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyf -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/2.5594e2

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/mpyf/2.5594e2/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyf 2.5594e2, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/7.8125e-3

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/mpyf/7.8125e-3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyf 7.8125e-3, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/mpyf/-7.8163e-3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 40-bit floating point register
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 mpyf -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/mpyf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/mpyf/-2.56e2/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 40-bit floating point register
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 mpyf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/mpyf
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/mpyf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 r2: a 40-bit floating point register
 ---- OUTPUTS ----
 r2: a 40-bit floating point register
 r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 mpyf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf **ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 cmpf *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 cmpf ***ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 cmpf *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 cmpf *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 cmpf *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpf *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 cmpf *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 cmpf **ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 cmpf *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpf *++ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 cmpf *--ar4(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpf *ar4++(ir0), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 cmpf *ar4--(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpf *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 cmpf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 cmpf *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 cmpf *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 cmpf *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 cmpf *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpf *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 cmpf *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 cmpf *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 cmpf *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 cmpf *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 cmpf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 cmpf *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 cmpf *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/cmpf/indir
Pattern: patterns/src_float_indir_src_float_reg.pat
Manually selected input: inputs/src_float_indir_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/cmpf/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_src_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_src_float_reg.txt (0 tests)
Random input: 2ops_float_indir/cmpf/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 cmpf *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_reg/cmpf
Pattern: patterns/2ops_src_float_reg_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_float_reg.txt (0 tests)
Random input: 2ops_float_reg/cmpf/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
ldiu r0, st
cmpf r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_imm/cmpf/0.0
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/0.0/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpf 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/1.0
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/1.0/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpf 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/-1.0
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/-1.0/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpf -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/1.5
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/1.5/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpf 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/-1.5
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/-1.5/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpf -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/2.5594e2/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpf 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/7.8125e-3/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
ldiu r0, st
cmpf 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/-7.8163e-3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 40-bit floating point register
 ---- OUTPUTS ----
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf -7.8163e-3, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/cmpf/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_float_reg.txt (0 tests)
Random input: 2ops_float_imm/cmpf/-2.56e2/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 r1: a 40-bit floating point register
 ---- OUTPUTS ----
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/cmpf
Pattern: patterns/src_float_dir_src_float_reg.pat
Manually selected input: inputs/src_float_dir_src_float_reg.txt (0 tests)
Random input: 2ops_float_dir/cmpf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 r2: a 40-bit floating point register
 ---- OUTPUTS ----
 r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 cmpf @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_predisp.add

Pattern: patterns/src_float_indir_predisp.add_dst_int_reg.pat

Manually selected input: inputs/src_float_indir_predisp.add_dst_int_reg.txt (0 tests)

Random input: 2ops_float_indir/fix/indir_predisp.add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

---- OUTPUTS ----

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu r0, st

fix *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_predisp.sub

Pattern: patterns/src_float_indir_predisp.sub_dst_int_reg.pat

Manually selected input: inputs/src_float_indir_predisp.sub_dst_int_reg.txt (0 tests)

Random input: 2ops_float_indir/fix/indir_predisp.sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

addi 16, ar2 go after the end of the source buffer

ldiu r0, st

fix *-ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_predisp.add_mod

Pattern: patterns/src_float_indir_predisp.add_mod_dst_int_reg.pat

Manually selected input: inputs/src_float_indir_predisp.add_mod_dst_int_reg.txt (0 tests)

Random input: 2ops_float_indir/fix/indir_predisp.add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit floating point values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 32-bit integer register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

fix *++ar4(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 fix *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 fix *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 fix *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_postdisp_sub_circ_mod_bk.5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postdisp_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 fix *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 fix *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 fix *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
and 15, r0 *crop random value between 0 and 15*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to the source buffer*
addi 15, ar4 *go at the end of the source buffer*
ldiu r1, st
fix *ar4--(ir0), r2 \leftarrow *instruction under test*
ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:

---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu 4, bk *load block size (should be at most 8)*
and 4 - 1, r0 *crop random value between 0 and bk - 1*
ldiu r0, ir0 *load ir0 with this random value*
ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
addi 7, ar4
andn 7, ar4 *align circular buffer start on block size*
ldiu r1, st
fix *ar4++(ir0)%, r2 \leftarrow *instruction under test*
ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 fix *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 fix *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 fix *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *++ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 fix *--ar4(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *ar4++(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 fix *ar4--(ir1), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4++(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 fix *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 fix *ar4++(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 fix *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix/indir
Pattern: patterns/src_float_indir_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 fix *+ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_bit_rev_mod_dst_int_reg.txt (0 tests)
Random input: 2ops_float_indir/fix/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 fix *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_predisp_add
Pattern: patterns/src_float_indir_predisp_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_predisp_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 fix *+ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_predisp_sub
Pattern: patterns/src_float_indir_predisp_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_predisp_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 addi 16, ar2 go after the end of the source buffer
 ldiu r0, st
 fix *-ar2(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_predisp_add_mod
Pattern: patterns/src_float_indir_predisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_predisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 fix *++ar4(1), r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_predisp_sub_mod
Pattern: patterns/src_float_indir_predisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_predisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 fix *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postdisp_add_mod
Pattern: patterns/src_float_indir_postdisp_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 fix *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postdisp_sub_mod
Pattern: patterns/src_float_indir_postdisp_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 fix *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postdisp_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 fix *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_add
Pattern: patterns/src_float_indir_preind_ir0_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 fix *+ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_sub
Pattern: patterns/src_float_indir_preind_ir0_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu r1, st
 fix *-ar2(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_add_mod
Pattern: patterns/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_sub_mod
Pattern: patterns/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 fix *--ar4(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add_mod
Pattern: patterns/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *ar4++(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_sub_mod
Pattern: patterns/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 fix *ar4--(ir0), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4++(ir0)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 and 4 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 fix *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 fix *ar4++(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 fix *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_add
Pattern: patterns/src_float_indir_preind_ir1_add_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 fix *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_sub
Pattern: patterns/src_float_indir_preind_ir1_sub_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 *go at the end of the source buffer*
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu r1, st
 fix *-ar2(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_add_mod
Pattern: patterns/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_sub_mod
Pattern: patterns/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_preind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 fix *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_add_mod
Pattern: patterns/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 fix *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_sub_mod
Pattern: patterns/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 fix *ar4--(ir1), r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4++(ir1)%, r2 \leftarrow *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_sub_circ_mod_bk_4
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_add_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_add_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_sub_circ_mod_bk_5
Pattern: patterns/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir1_sub_circ_mod_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir1_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 fix *ar4--(ir1)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_dst_ext/indir
Pattern: patterns/src_float_indir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 fix *+ar2, r1 *← instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add.bit_rev_mod
Pattern: patterns/src_float_indir_postind_ir0_add.bit_rev_mod_src_dst_float_reg.pat
Manually selected input: inputs/src_float_indir_postind_ir0_add.bit_rev_mod_src_dst_float_reg.
↳ txt (0 tests)
Random input: 2ops_float_indir/fix_dst_ext/indir_postind_ir0_add.bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit floating point values
r1: a 32-bit integer register (value for st)
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
and 15, r0 crop random value between 0 and 15
ldiu r0, ir0 load ir0 with this random value
ldiu ar2, ar4 load a pointer to the buffer
ldiu r1, st
fix *ar4++(ir0)b, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_reg/fix
Pattern: patterns/2ops_src_float_reg_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_reg_dst_int_reg.txt (0 tests)
Random input: 2ops_float_reg/fix/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r2: a 32-bit integer register
r3: a 32-bit integer register (value of st)
ldiu r0, st
fix r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_reg/fix_dst_ext
Pattern: patterns/2ops_src_float_reg_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_reg_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/fix_dst_ext/random.txt (10000 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
fix r1, r2 ← instruction under test
ldiu st, r3

Subdirectory: 2ops_float_imm/fix/0.0
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/0.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/1.0
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/-1.0
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/-1.0/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix -1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/1.5
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/-1.5
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/-1.5/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix -1.5, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/2.5594e2
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/2.5594e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/7.8125e-3
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/7.8125e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/-7.8163e-3/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix/-2.56e2
Pattern: patterns/2ops_src_float_imm_dst_int_reg.pat
Manually selected input: inputs/2ops_src_float_imm_dst_int_reg.txt (0 tests)
Random input: 2ops_float_imm/fix/-2.56e2/random.txt (1 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
---- OUTPUTS ----
r1: a 32-bit integer register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix -2.56e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/0.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/fix_dst_ext/0.0/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 0.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/1.0
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/fix_dst_ext/1.0/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 1.0, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/-1.0

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/fix_dst_ext/-1.0/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

fix -1.0, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/fix_dst_ext/1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

fix 1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/-1.5

Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat

Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)

Random input: 2ops_float_imm/fix_dst_ext/-1.5/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

fix -1.5, r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/2.5594e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/fix_dst_ext/2.5594e2/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 2.5594e2, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/7.8125e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/fix_dst_ext/7.8125e-3/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix 7.8125e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/-7.8163e-3
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/fix_dst_ext/-7.8163e-3/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
fix -7.8163e-3, r1 ← instruction under test
ldiu st, r2

Subdirectory: 2ops_float_imm/fix_dst_ext/-2.56e2
Pattern: patterns/2ops_src_float_imm_src_dst_float_reg.pat
Manually selected input: inputs/2ops_src_float_imm_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/fix_dst_ext/-2.56e2/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 fix -2.56e2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/fix
Pattern: patterns/src_float_dir_src_dst_float_reg.pat
Manually selected input: inputs/src_float_dir_src_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/fix/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldfu *ar2, r1 load input value
 stf r1, @_input store input value into _input
 fix @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/fix_ar_update_ordering
Pattern: patterns/2ops_src_float_buf_dst_int_reg_ar_update_ordering.pat
Manually selected input: inputs/2ops_src_float_buf_dst_int_reg_ar_update_ordering.txt (0 tests)
Random input: 2ops_float_indir/fix_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 fix *ar4++(1), ar4 ← instruction under test
 ldiu st, r1

Subdirectory: 2ops_float_indir/float/indir_predisp_add

Pattern: patterns/src_int_indir_predisp_add_dst_float_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/float/indir_predisp_add/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

float *+ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_predisp_sub

Pattern: patterns/src_int_indir_predisp_sub_dst_float_reg.pat

Manually selected input: inputs/src_int_indir_predisp_sub_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/float/indir_predisp_sub/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

addi 16, ar2 go after the end of the source buffer

ldiu r0, st

float *-ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_predisp_add_mod

Pattern: patterns/src_int_indir_predisp_add_mod_dst_float_reg.pat

Manually selected input: inputs/src_int_indir_predisp_add_mod_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/float/indir_predisp_add_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r0: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu ar2, ar4

ldiu r0, st

float *++ar4(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_predisp_sub_mod
Pattern: patterns/src_int_indir_predisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_predisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_predisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 16, ar4 *go after the end of the source buffer*
 ldiu r0, st
 float *--ar4(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_postdisp_add_mod
Pattern: patterns/src_int_indir_postdisp_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postdisp_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 ldiu r0, st
 float *ar4++(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_postdisp_sub_mod
Pattern: patterns/src_int_indir_postdisp_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postdisp_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu ar2, ar4
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r0, st
 float *ar4--(1), r1 ← *instruction under test*
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_postdisp_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postdisp_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 float *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_postdisp_sub_circ_mod_bk_4
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postdisp_sub_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 4, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 float *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_postdisp_add_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postdisp_add_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 float *ar4++(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_postdisp_sub_circ_mod_bk_5
Pattern: patterns/src_int_indir_postdisp_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postdisp_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postdisp_sub_circ_mod_bk_5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r0, st
 float *ar4--(1)%, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_preind_ir0_add
Pattern: patterns/src_int_indir_preind_ir0_add_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir0_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 float *+ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_preind_ir0_sub
Pattern: patterns/src_int_indir_preind_ir0_sub_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir0_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu r1, st
 float *-ar2(ir0), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_preind_ir0_add_mod
Pattern: patterns/src_int_indir_preind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 float *++ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_preind_ir0_sub_mod
Pattern: patterns/src_int_indir_preind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir0_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 float *--ar4(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir0_add_mod
Pattern: patterns/src_int_indir_postind_ir0_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir0_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 float *ar4++(ir0), r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir0_sub_mod
Pattern: patterns/src_int_indir_postind_ir0_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir0_sub_mod/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

and 15, r0 *crop random value between 0 and 15*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to the source buffer*

addi 15, ar4 *go at the end of the source buffer*

ldiu r1, st

float *ar4--(ir0), r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir0_add_circ_mod_bk_4

Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_float_reg.pat

Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)

Random input: 2ops_float_indir/float/indir_postind_ir0_add_circ_mod_bk_4/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register

ar2: an auxiliary register pointing to an array of 16 32-bit integer values

r1: a 32-bit integer register (value for st)

---- OUTPUTS ----

ar4: an auxiliary register

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu 4, bk *load block size (should be at most 8)*

and 4 - 1, r0 *crop random value between 0 and bk - 1*

ldiu r0, ir0 *load ir0 with this random value*

ldiu ar2, ar4 *load a pointer to a buffer of 16 words*

addi 7, ar4

andn 7, ar4 *align circular buffer start on block size*

ldiu r1, st

float *ar4++(ir0)%, r2 *← instruction under test*

ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir0_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir0_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 float *ar4--(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir0_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postind_ir0_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir0_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir0 *load ir0 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 float *ar4++(ir0)%, r2 *← instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir0_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postind_ir0_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir0_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 float *ar4--(ir0)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_preind_ir1_add
Pattern: patterns/src_int_indir_preind_ir1_add_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir1_add/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 float *+ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_preind_ir1_sub
Pattern: patterns/src_int_indir_preind_ir1_sub_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir1_sub/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r0: a 32-bit integer register
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 addi 15, ar2 go at the end of the source buffer
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir1 load ir1 with this random value
 ldiu r1, st
 float *-ar2(ir1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_preind_ir1_add_mod
Pattern: patterns/src_int_indir_preind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 float *++ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_preind_ir1_sub_mod
Pattern: patterns/src_int_indir_preind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_preind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_preind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 16, ar4 *go just after the end of the source buffer*
 ldiu r1, st
 float *--ar4(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir1_add_mod
Pattern: patterns/src_int_indir_postind_ir1_add_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir1_add_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the buffer*
 ldiu r1, st
 float *ar4++(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir1_sub_mod
Pattern: patterns/src_int_indir_postind_ir1_sub_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir1_sub_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 *crop random value between 0 and 15*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to the source buffer*
 addi 15, ar4 *go at the end of the source buffer*
 ldiu r1, st
 float *ar4--(ir1), r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir1_add_circ_mod_bk_4
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir1_add_circ_mod_bk_4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 float *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir1_sub_circ_mod_bk.4
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir1_sub_circ_mod_bk.4/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 4, bk *load block size (should be at most 8)*
 and 4 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 float *ar4--(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir1_add_circ_mod_bk.5
Pattern: patterns/src_int_indir_postind_ir1_add_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_add_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir1_add_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk *load block size (should be at most 8)*
 and 5 - 1, r0 *crop random value between 0 and bk - 1*
 ldiu r0, ir1 *load ir1 with this random value*
 ldiu ar2, ar4 *load a pointer to a buffer of 16 words*
 addi 7, ar4
 andn 7, ar4 *align circular buffer start on block size*
 ldiu r1, st
 float *ar4++(ir1)%, r2 ← *instruction under test*
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir_postind_ir1_sub_circ_mod_bk.5
Pattern: patterns/src_int_indir_postind_ir1_sub_circ_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir1_sub_circ_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir1_sub_circ_mod_bk.5/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu 5, bk load block size (should be at most 8)
 and 5 - 1, r0 crop random value between 0 and bk - 1
 ldiu r0, ir1 load ir1 with this random value
 ldiu ar2, ar4 load a pointer to a buffer of 16 words
 addi 7, ar4
 andn 7, ar4 align circular buffer start on block size
 ldiu r1, st
 float *ar4--(ir1)%, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_indir/float/indir
Pattern: patterns/src_int_indir_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 float **ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_indir/float/indir_postind_ir0_add_bit_rev_mod
Pattern: patterns/src_int_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.pat
Manually selected input: inputs/src_int_indir_postind_ir0_add_bit_rev_mod_dst_float_reg.txt (0 tests)
Random input: 2ops_float_indir/float/indir_postind_ir0_add_bit_rev_mod/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register
ar2: an auxiliary register pointing to an array of 16 32-bit integer values
r1: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
ar4: an auxiliary register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 and 15, r0 crop random value between 0 and 15
 ldiu r0, ir0 load ir0 with this random value
 ldiu ar2, ar4 load a pointer to the buffer
 ldiu r1, st
 float *ar4++(ir0)b, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_reg/float
Pattern: patterns/2ops_src_int_reg_dst_float_reg.pat
Manually selected input: inputs/2ops_src_int_reg_dst_float_reg.txt (0 tests)
Random input: 2ops_float_reg/float/random.txt (10000 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
 ---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 float r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 2ops_float_imm/float/0
Pattern: patterns/2ops_src_int_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/float/0/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 float 0, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/float/1
Pattern: patterns/2ops_src_int_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/float/1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 float 1, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/float/-1
Pattern: patterns/2ops_src_int_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/float/-1/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 float -1, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/float/-32768
Pattern: patterns/2ops_src_int_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/float/-32768/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 float -32768, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_imm/float/32767
Pattern: patterns/2ops_src_int_imm_dst_float_reg.pat
Manually selected input: inputs/2ops_src_int_imm_dst_float_reg.txt (0 tests)
Random input: 2ops_float_imm/float/32767/random.txt (1 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ---- OUTPUTS ----
 r1: a 40-bit floating point register
 r2: a 32-bit integer register (value of st)
 ldiu r0, st
 float 32767, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 2ops_float_dir/float
Pattern: patterns/src_int_dir_dst_float_reg.pat
Manually selected input: inputs/src_int_dir_dst_float_reg.txt (0 tests)
Random input: 2ops_float_dir/float/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
 r0: a 32-bit integer register (value for st)
 ar2: an auxiliary register pointing to an array of 1 32-bit integer values
 ---- OUTPUTS ----
 r2: a 40-bit floating point register
 r3: a 32-bit integer register (value of st)
 .data
 _input .word 55555555h input value
 .text
 ldiu r0, st
 ldiu *ar2, r1 load input value
 sti r1, @_input store input value into _input
 float @_input, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_float_reg_reg/addf3

Pattern: patterns/3ops_src_float_reg_src_float_reg_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)

Random input: 3ops_float_reg_reg/addf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r2: a 40-bit floating point register

---- OUTPUTS ----

r3: a 40-bit floating point register

r4: a 32-bit integer register (value of st)

ldiu r0, st

addf3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_float_reg_buf/addf3

Pattern: patterns/3ops_src_float_reg_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_reg_src_float_buf_dst_float_reg.txt (0 tests)

Random input: 3ops_float_reg_buf/addf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu r0, st

addf3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_float_buf_reg/addf3

Pattern: patterns/3ops_src_float_buf_src_float_reg_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_reg_dst_float_reg.txt (0 tests)

Random input: 3ops_float_buf_reg/addf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r1: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu r0, st

addf3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_float_buf_buf/addf3

Pattern: patterns/3ops_src_float_buf_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_buf_dst_float_reg.txt (0 tests)

Random input: 3ops_float_buf_buf/addf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit floating point values

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

addf3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_float_buf_buf/addf3_ar_update_ordering

Pattern: patterns/3ops_src_float_buf_src_float_buf_dst_float_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_buf_dst_float_reg_ar_update_ordering.
↩ txt (0 tests)

Random input: 3ops_float_buf_buf/addf3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

addf3 *ar4++(1), *ar4--(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_float_reg_reg/subf3

Pattern: patterns/3ops_src_float_reg_src_float_reg_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)

Random input: 3ops_float_reg_reg/subf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r2: a 40-bit floating point register

---- OUTPUTS ----

r3: a 40-bit floating point register

r4: a 32-bit integer register (value of st)

ldiu r0, st

subf3 r1, r2, r3 ← instruction under test

ldiu st, r4

Subdirectory: 3ops_float_reg_buf/subf3

Pattern: patterns/3ops_src_float_reg_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_reg_src_float_buf_dst_float_reg.txt (0 tests)

Random input: 3ops_float_reg_buf/subf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subf3 r1, *ar2, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_float_buf_reg/subf3

Pattern: patterns/3ops_src_float_buf_src_float_reg_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_reg_dst_float_reg.txt (0 tests)

Random input: 3ops_float_buf_reg/subf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r1: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu r0, st

subf3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_float_buf_buf/subf3

Pattern: patterns/3ops_src_float_buf_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_buf_dst_float_reg.txt (0 tests)

Random input: 3ops_float_buf_buf/subf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit floating point values

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

subf3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_float_buf_buf/subf3_ar_update_ordering
Pattern: patterns/3ops_src_float_buf_src_float_buf_dst_float_reg_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_float_buf_src_float_buf_dst_float_reg_ar_update_ordering.
↳ txt (0 tests)
Random input: 3ops_float_buf_buf/subf3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 40-bit floating point register
r2: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar4, ar5
subf3 *ar4++(1), *ar4--(1), r1 ← instruction under test
ldiu st, r2

Subdirectory: 3ops_float_reg_reg/mpyf3
Pattern: patterns/3ops_src_float_reg_src_float_reg_dst_float_reg.pat
Manually selected input: inputs/3ops_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)
Random input: 3ops_float_reg_reg/mpyf3/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 32-bit integer register (value of st)
ldiu r0, st
mpyf3 r1, r2, r3 ← instruction under test
ldiu st, r4

Subdirectory: 3ops_float_reg_buf/mpyf3
Pattern: patterns/3ops_src_float_reg_src_float_buf_dst_float_reg.pat
Manually selected input: inputs/3ops_src_float_reg_src_float_buf_dst_float_reg.txt (0 tests)
Random input: 3ops_float_reg_buf/mpyf3/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
---- OUTPUTS ----
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
mpyf3 r1, *ar2, r2 ← instruction under test
ldiu st, r3

Subdirectory: 3ops_float_buf_reg/mpyf3

Pattern: patterns/3ops_src_float_buf_src_float_reg_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_reg_dst_float_reg.txt (0 tests)

Random input: 3ops_float_buf_reg/mpyf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r1: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 32-bit integer register (value of st)

ldiu r0, st

mpyf3 *ar2, r1, r2 ← instruction under test

ldiu st, r3

Subdirectory: 3ops_float_buf_buf/mpyf3

Pattern: patterns/3ops_src_float_buf_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_buf_dst_float_reg.txt (0 tests)

Random input: 3ops_float_buf_buf/mpyf3/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 2 32-bit floating point values

---- OUTPUTS ----

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

mpyf3 *ar2, **ar2(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_float_buf_buf/mpyf3_ar_update_ordering

Pattern: patterns/3ops_src_float_buf_src_float_buf_dst_float_reg_ar_update_ordering.pat

Manually selected input: inputs/3ops_src_float_buf_src_float_buf_dst_float_reg_ar_update_ordering.
↩ txt (0 tests)

Random input: 3ops_float_buf_buf/mpyf3_ar_update_ordering/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 40-bit floating point register

r2: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar4, ar5

mpyf3 *ar4++(1), *ar4--(1), r1 ← instruction under test

ldiu st, r2

Subdirectory: 3ops_float_reg_reg/cmpf3
Pattern: patterns/3ops_src_float_reg_src_float_reg.pat
Manually selected input: inputs/3ops_src_float_reg_src_float_reg.txt (0 tests)
Random input: 3ops_float_reg_reg/cmpf3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf3 r1, r2 ← instruction under test
 ldiu st, r3

Subdirectory: 3ops_float_reg_buf/cmpf3
Pattern: patterns/3ops_src_float_reg_src_float_buf.pat
Manually selected input: inputs/3ops_src_float_reg_src_float_buf.txt (0 tests)
Random input: 3ops_float_reg_buf/cmpf3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf3 r1, *ar2 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_float_buf_reg/cmpf3
Pattern: patterns/3ops_src_float_buf_src_float_reg.pat
Manually selected input: inputs/3ops_src_float_buf_src_float_reg.txt (0 tests)
Random input: 3ops_float_buf_reg/cmpf3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
 ---- OUTPUTS ----
r2: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf3 *ar2, r1 ← instruction under test
 ldiu st, r2

Subdirectory: 3ops_float_buf_buf/cmpf3
Pattern: patterns/3ops_src_float_buf_src_float_buf.pat
Manually selected input: inputs/3ops_src_float_buf_src_float_buf.txt (0 tests)
Random input: 3ops_float_buf_buf/cmpf3/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 2 32-bit floating point values
 ---- OUTPUTS ----
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 cmpf3 *ar2, **ar2(1) ← instruction under test
 ldiu st, r1

Subdirectory: 3ops_float_buf_buf/cmpf3_ar_update_ordering
Pattern: patterns/3ops_src_float_buf_src_float_buf_ar_update_ordering.pat
Manually selected input: inputs/3ops_src_float_buf_src_float_buf_ar_update_ordering.txt (0 tests)
Random input: 3ops_float_buf_buf/cmpf3_ar_update_ordering/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar4, ar5
 cmpf3 *ar4++(1), *ar4--(1) ← instruction under test
 ldiu st, r1

Subdirectory: parallel_float/ldf_ldf/buf_buf
Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_buf_dst_float_reg.pat
Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_buf_dst_float_reg.txt (0 tests)
Random input: parallel_float/ldf_ldf/buf_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
ar4: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r1: a 40-bit floating point register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldf *ar2, r1
 || ldf *ar4, r2
 ldiu st, r3

Subdirectory: parallel_float/ldf_ldf_ar_update_ordering/buf_buf
Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_buf_dst_float_reg_ar_update_ordering.
 ↪ pat
Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_buf_dst_float_reg_ar_
 ↪ update_ordering.txt (0 tests)
Random input: parallel_float/ldf_ldf_ar_update_ordering/buf_buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 40-bit floating point register
r2: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 ldf *ar4++(1), r1
 || ldf *ar4--(1), r2 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_float/ldf_ldf/reg_buf

Pattern: patterns/par_src_float_reg_dst_float_reg_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/par_src_float_reg_dst_float_reg_src_float_buf_dst_float_reg.txt (0 tests)

Random input: parallel_float/ldf_ldf/reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r2: a 40-bit floating point register

r3: a 40-bit floating point register

r4: a 32-bit integer register (value of st)

ldiu r0, st

ldf r1, r2

|| ldf *ar2, r3

ldiu st, r4

Subdirectory: parallel_float/ldf_stf/buf_buf

Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)

Random input: parallel_float/ldf_stf/buf_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r2: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldf *ar2, r1

|| stf r2, *ar4 ← instruction under test

ldiu st, r3

Subdirectory: parallel_float/ldf_stf_ar.update_ordering/buf_buf
Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar.update_ordering.
↪ pat
Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_
↪ update_ordering.txt (0 tests)
Random input: parallel_float/ldf_stf_ar.update_ordering/buf_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
ldf *ar4++(1), r1
|| stf r2, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_float/ldf_stf/reg_buf
Pattern: patterns/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.pat
Manually selected input: inputs/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)
Random input: parallel_float/ldf_stf/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r3: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 32-bit integer register (value of st)
ldiu r0, st
ldf r1, r2
|| stf r3, *ar2 ← instruction under test
ldiu st, r4

Subdirectory: parallel_float/stf_stf/buf_buf
Pattern: patterns/par_src_float_reg_dst_float_buf_src_float_reg_dst_float_buf.pat
Manually selected input: inputs/par_src_float_reg_dst_float_buf_src_float_reg_dst_float_buf.txt (0 tests)
Random input: parallel_float/stf_stf/buf_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r2: a 40-bit floating point register
---- OUTPUTS ----
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
ar4: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 32-bit integer register (value of st)
ldiu r0, st
stf r1, *ar2
|| stf r2, *ar4 ← instruction under test
ldiu st, r3

Subdirectory: parallel_float/stf_stf_ar.update.ordering/buf_buf
Pattern: patterns/par_src_float_reg_dst_float_buf_src_float_reg_dst_float_buf_ar.update.ordering.
↪ pat
Manually selected input: inputs/par_src_float_reg_dst_float_buf_src_float_reg_dst_float_buf_ar_
↪ update_ordering.txt (0 tests)
Random input: parallel_float/stf_stf_ar.update.ordering/buf_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r1: a 40-bit floating point register
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
stf r1, *ar4++(1)
|| stf r2, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_float/stf_stf/reg_buf
Pattern: patterns/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.pat
Manually selected input: inputs/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)
Random input: parallel_float/stf_stf/reg_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r3: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 32-bit integer register (value of st)
ldiu r0, st
stf r1, r2
|| stf r3, *ar2 ← instruction under test
ldiu st, r4

Subdirectory: parallel_float/fix_sti/buf_reg

Pattern: patterns/par_src_float_buf_src_dst_float_reg_src_float_reg_dst_int_buf.pat

Manually selected input: inputs/par_src_float_buf_src_dst_float_reg_src_float_reg_dst_int_buf.

↪ txt (0 tests)

Random input: parallel_float/fix_sti/buf_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r1: a 40-bit floating point register

r2: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit integer values

r3: a 32-bit integer register (value of st)

ldiu r0, st

fix *ar2, r1

|| sti r2, *ar4 ← instruction under test

ldiu st, r3

Subdirectory: parallel_float/fix_sti_ar_update_ordering/buf_reg

Pattern: patterns/par_src_float_buf_dst_int_reg_src_float_reg_dst_int_buf_ar_update_ordering.pat

Manually selected input: inputs/par_src_float_buf_dst_int_reg_src_float_reg_dst_int_buf_ar_update_

↪ ordering.txt (0 tests)

Random input: parallel_float/fix_sti_ar_update_ordering/buf_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r2: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r1: a 32-bit integer register

r3: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar2, ar5

fix *ar4++(1), r1

|| sti r2, *ar4--(1) ← instruction under test

ldiu st, r3

Subdirectory: parallel_float/fix_sti/reg_reg

Pattern: patterns/par_src_float_reg_src_dst_float_reg_src_float_reg_dst_int_buf.pat

Manually selected input: inputs/par_src_float_reg_src_dst_float_reg_src_float_reg_dst_int_buf.

↪ txt (0 tests)

Random input: parallel_float/fix_sti/reg_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r2: a 40-bit floating point register

r3: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r4: a 32-bit integer register (value of st)

ldiu r0, st

fix r1, r2

|| sti r3, *ar2 ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/float_stf/buf_reg

Pattern: patterns/par_src_int_buf_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_int_buf_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)

Random input: parallel_float/float_stf/buf_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit integer values

r2: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 32-bit integer register (value of st)

ldiu r0, st

float *ar2, r1

|| stf r2, *ar4 ← instruction under test

ldiu st, r3

Subdirectory: parallel_float/float_stf_ar_update_ordering/buf_reg
Pattern: patterns/par_src_int_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_update_ordering.
↳ pat
Manually selected input: inputs/par_src_int_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_
↳ update_ordering.txt (0 tests)
Random input: parallel_float/float_stf_ar_update_ordering/buf_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit integer values
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
float *ar4++(1), r1
|| stf r2, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_float/float_stf/reg_reg
Pattern: patterns/par_src_int_reg_dst_float_reg_src_float_reg_dst_float_buf.pat
Manually selected input: inputs/par_src_int_reg_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)
Random input: parallel_float/float_stf/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 32-bit integer register
r3: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 32-bit integer register (value of st)
ldiu r0, st
float r1, r2
|| stf r3, *ar2 ← instruction under test
ldiu st, r4

Subdirectory: parallel_float/absf_stf/buf_reg
Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.pat
Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)
Random input: parallel_float/absf_stf/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
r1: a 40-bit floating point register
ar4: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 absf *ar2, r1
 || stf r2, *ar4 ← instruction under test
 ldiu st, r3

Subdirectory: parallel_float/absf_stf_ar.update_ordering/buf_reg
Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar.update_ordering.
 ↩ pat
Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_
 ↩ update_ordering.txt (0 tests)
Random input: parallel_float/absf_stf_ar.update_ordering/buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
 ldiu r0, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 absf *ar4++(1), r1
 || stf r2, *ar4--(1) ← instruction under test
 ldiu st, r3

Subdirectory: parallel_float/absf_stf/reg_reg

Pattern: patterns/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)

Random input: parallel_float/absf_stf/reg_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r3: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r4: a 32-bit integer register (value of st)

ldiu r0, st

absf r1, r2

|| stf r3, *ar2 ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/negf_stf/buf_reg

Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)

Random input: parallel_float/negf_stf/buf_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r2: a 40-bit floating point register

---- OUTPUTS ----

r1: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 32-bit integer register (value of st)

ldiu r0, st

negf *ar2, r1

|| stf r2, *ar4 ← instruction under test

ldiu st, r3

Subdirectory: parallel_float/negf_stf_ar_update_ordering/buf_reg
Pattern: patterns/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_update_ordering.
↳ pat
Manually selected input: inputs/par_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_
↳ update_ordering.txt (0 tests)
Random input: parallel_float/negf_stf_ar_update_ordering/buf_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r2: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r1: a 40-bit floating point register
r3: a 32-bit integer register (value of st)
ldiu r0, st
ldiu ar2, ar4
ldiu ar2, ar5
negf *ar4++(1), r1
|| stf r2, *ar4--(1) ← instruction under test
ldiu st, r3

Subdirectory: parallel_float/negf_stf/reg_reg
Pattern: patterns/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.pat
Manually selected input: inputs/par_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)
Random input: parallel_float/negf_stf/reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r0: a 32-bit integer register (value for st)
r1: a 40-bit floating point register
r3: a 40-bit floating point register
---- OUTPUTS ----
r2: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 32-bit integer register (value of st)
ldiu r0, st
negf r1, r2
|| stf r3, *ar2 ← instruction under test
ldiu st, r4

Subdirectory: parallel_float/addf3.stf/reg.buf

Pattern: patterns/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)

Random input: parallel_float/addf3.stf/reg.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

r4: a 32-bit integer register (value of st)

ldiu r0, st

addf3 r1, *ar2, r2

|| stf r3, *ar4 ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/addf3.stf_ar_update_ordering/reg.buf

Pattern: patterns/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_update_ordering.pat

Manually selected input: inputs/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_update_ordering.txt (0 tests)

Random input: parallel_float/addf3.stf_ar_update_ordering/reg.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r1: a 40-bit floating point register

r3: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r2: a 40-bit floating point register

r4: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar2, ar5

addf3 r1, *ar4++(1), r2

|| stf r3, *ar4--(1) ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/addf3.stf/reg.reg

Pattern: patterns/par_src.float_reg_src.float_reg_dst.float_reg_src.float_reg_dst.float_buf.pat

Manually selected input: inputs/par_src.float_reg_src.float_reg_dst.float_reg_src.float_reg_dst_↵ float_buf.txt (0 tests)

Random input: parallel_float/addf3.stf/reg.reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r2: a 40-bit floating point register

r4: a 40-bit floating point register

---- OUTPUTS ----

r3: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r5: a 32-bit integer register (value of st)

ldiu r0, st

addf3 r1, r2, r3

|| stf r4, *ar2 ← instruction under test

ldiu st, r5

Subdirectory: parallel_float/subf3.stf/reg.buf

Pattern: patterns/par_src.float_reg_src.float_buf_dst.float_reg_src.float_reg_dst.float_buf.pat

Manually selected input: inputs/par_src.float_reg_src.float_buf_dst.float_reg_src.float_reg_dst_↵ float_buf.txt (0 tests)

Random input: parallel_float/subf3.stf/reg.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

r4: a 32-bit integer register (value of st)

ldiu r0, st

subf3 r1, *ar2, r2

|| stf r3, *ar4 ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/subf3_stf_ar.update_ordering/reg.buf

Pattern: patterns/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_↪
↪ update_ordering.pat

Manually selected input: inputs/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_↪
↪ float_buf_ar.update_ordering.txt (0 tests)

Random input: parallel_float/subf3_stf_ar.update_ordering/reg.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r1: a 40-bit floating point register

r3: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r2: a 40-bit floating point register

r4: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar2, ar5

subf3 r1, *ar4++(1), r2

|| stf r3, *ar4--(1) ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/subf3_stf/reg.reg

Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_dst_↪
↪ float_buf.txt (0 tests)

Random input: parallel_float/subf3_stf/reg.reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r2: a 40-bit floating point register

r4: a 40-bit floating point register

---- OUTPUTS ----

r3: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r5: a 32-bit integer register (value of st)

ldiu r0, st

subf3 r1, r2, r3

|| stf r4, *ar2 ← instruction under test

ldiu st, r5

Subdirectory: parallel_float/mpyf3.stf/reg.buf

Pattern: patterns/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)

Random input: parallel_float/mpyf3.stf/reg.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

---- OUTPUTS ----

r2: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

r4: a 32-bit integer register (value of st)

ldiu r0, st

mpyf3 r1, *ar2, r2

|| stf r3, *ar4 ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/mpyf3.stf_ar_update_ordering/reg.buf

Pattern: patterns/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_update_ordering.pat

Manually selected input: inputs/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_dst_float_buf_ar_update_ordering.txt (0 tests)

Random input: parallel_float/mpyf3.stf_ar_update_ordering/reg.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r1: a 40-bit floating point register

r3: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r2: a 40-bit floating point register

r4: a 32-bit integer register (value of st)

ldiu r0, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyf3 r1, *ar4++(1), r2

|| stf r3, *ar4--(1) ← instruction under test

ldiu st, r4

Subdirectory: parallel_float/mpyf3.stf/reg_reg

Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.pat

Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_dst_float_buf.txt (0 tests)

Random input: parallel_float/mpyf3.stf/reg_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r0: a 32-bit integer register (value for st)

r1: a 40-bit floating point register

r2: a 40-bit floating point register

r4: a 40-bit floating point register

---- OUTPUTS ----

r3: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r5: a 32-bit integer register (value of st)

ldiu r0, st

mpyf3 r1, r2, r3

|| stf r4, *ar2 ← instruction under test

ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3/buf_buf_reg_reg

Pattern: patterns/par_src_float_buf_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.pat

Manually selected input: inputs/par_src_float_buf_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)

Random input: parallel_float/mpyf3.addf3/buf_buf_reg_reg/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

r4: a 40-bit floating point register

---- OUTPUTS ----

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r5: a 32-bit integer register (value of st)

ldiu r1, st

mpyf3 *ar2, *ar4, r0

|| addf3 r3, r4, r2 ← instruction under test

ldfu r0, r3

ldfu r2, r4

ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3.ar.update.ordering/buf_buf_reg_reg
Pattern: patterns/par_src_float_buf_src_float_dst_float_reg_src_float_reg_src_float_reg_dst_↪ float_reg.ar.update.ordering.pat
Manually selected input: inputs/par_src_float_buf_src_float_buf_dst_float_reg_src_float_reg_src_↪ float_reg.dst.float_reg.ar.update.ordering.txt (0 tests)
Random input: parallel_float/mpyf3.addf3.ar.update.ordering/buf_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyf3 *ar4++(1), *ar4--(1), r0
 || addf3 r3, r4, r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r4
 ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3/buf_reg_reg_reg
Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_reg_dst_↪ float_reg.pat
Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_↪ float_reg.dst.float_reg.txt (0 tests)
Random input: parallel_float/mpyf3.addf3/buf_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 40-bit floating point register
 ---- OUTPUTS ----
r4: a 40-bit floating point register
r5: a 40-bit floating point register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyf3 *ar2, r3, r0
 || addf3 r4, r5, r2 ← instruction under test
 ldfu r0, r4
 ldfu r2, r5
 ldiu st, r6

Subdirectory: parallel_float/mpyf3.addf3/reg_buf_reg_reg
Pattern: patterns/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)
↪ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.addf3/reg_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 40-bit floating point register
r5: a 40-bit floating point register
---- OUTPUTS ----
r4: a 40-bit floating point register
r5: a 40-bit floating point register
r6: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 r3, *ar2, r0
|| addf3 r4, r5, r2 ← instruction under test
ldfu r0, r4
ldfu r2, r5
ldiu st, r6

Subdirectory: parallel_float/mpyf3.addf3/reg_reg_reg_reg
Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)
↪ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.addf3/reg_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
r4: a 40-bit floating point register
---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 r3, r4, r0
|| addf3 r3, r4, r2 ← instruction under test
ldfu r0, r3
ldfu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3/buf_reg.buf_reg
Pattern: patterns/par_src_float.buf_src_float_reg.dst_float_reg_src_float.buf_src_float_reg.dst_float_reg.pat
↳ float_reg.pat
Manually selected input: inputs/par_src_float.buf_src_float_reg.dst_float_reg_src_float.buf_src_float_reg.dst_float_reg.txt (0 tests)
↳ float_reg.dst_float_reg.txt
Random input: parallel_float/mpyf3.addf3/buf_reg.buf_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
ar4: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 40-bit floating point register
---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 *ar2, r3, r0
|| addf3 *ar4, r4, r2 ← instruction under test
ldfu r0, r3
ldfu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3.ar_update_ordering/buf_reg.buf_reg
Pattern: patterns/par_src_float.buf_src_float_reg.dst_float_reg_src_float.buf_src_float_reg.dst_float_reg.ar_update_ordering.pat
↳ float_reg.ar_update_ordering.pat
Manually selected input: inputs/par_src_float.buf_src_float_reg.dst_float_reg_src_float.buf_src_float_reg.dst_float_reg.ar_update_ordering.txt (0 tests)
↳ float_reg.dst_float_reg.ar_update_ordering.txt
Random input: parallel_float/mpyf3.addf3.ar_update_ordering/buf_reg.buf_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
ldiu ar2, ar4
ldiu ar2, ar5
mpyf3 *ar4++(1), r3, r0
|| addf3 *ar4--(1), r4, r2 ← instruction under test
ldiu r0, r3
ldiu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3/reg_reg.buf_reg
Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.txt (0 tests)
↪ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.addf3/reg_reg.buf_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
r4: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r5: a 40-bit floating point register
---- OUTPUTS ----
r4: a 40-bit floating point register
r5: a 40-bit floating point register
r6: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 r3, r4, r0
|| addf3 *ar2, r5, r2 ← instruction under test
ldfu r0, r4
ldfu r2, r5
ldiu st, r6

Subdirectory: parallel_float/mpyf3.addf3/reg_reg.buf_buf
Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_buf_src_float_buf_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_buf_src_float_buf_dst_float_reg.txt (0 tests)
↪ float_buf_dst_float_reg.txt
Random input: parallel_float/mpyf3.addf3/reg_reg.buf_buf/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
r4: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
ar4: an auxiliary register pointing to an array of 1 32-bit floating point values
---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 r3, r4, r0
|| addf3 *ar2, *ar4, r2 ← instruction under test
ldfu r0, r3
ldfu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3.ar.update.ordering/reg_reg.buf.buf
Pattern: patterns/par_src.float.reg_src.float.reg_dst.float.reg_src.float.buf_src.float.buf_dst_↪ float.reg.ar.update.ordering.pat
Manually selected input: inputs/par_src.float.reg_src.float.reg_dst.float.reg_src.float.buf_src_↪ float.buf_dst.float.reg.ar.update.ordering.txt (0 tests)
Random input: parallel_float/mpyf3.addf3.ar.update.ordering/reg_reg.buf.buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyf3 r3, r4, r0
 || addf3 *ar4++(1), *ar4--(1), r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r4
 ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3/reg_reg.reg.buf
Pattern: patterns/par_src.float.reg_src.float.reg_dst.float.reg_src.float.reg_src.float.buf_dst_↪ float.reg.pat
Manually selected input: inputs/par_src.float.reg_src.float.reg_dst.float.reg_src.float.reg_src_↪ float.buf_dst.float.reg.txt (0 tests)
Random input: parallel_float/mpyf3.addf3/reg_reg.reg.buf/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
 ---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyf3 r3, r4, r0
 || addf3 r5, *ar2, r2 ← instruction under test
 ldfu r0, r3
 ldfu r2, r4
 ldiu st, r6

Subdirectory: parallel_float/mpyf3.addf3/buf_reg_reg_buf

Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg.txt (0 tests)

Random input: parallel_float/mpyf3.addf3/buf_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

r4: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r5: a 32-bit integer register (value of st)

ldiu r1, st

mpyf3 *ar2, r3, r0

|| addf3 r4, *ar4, r2 ← instruction under test

ldfu r0, r3

ldfu r2, r4

ldiu st, r5

Subdirectory: parallel_float/mpyf3.addf3_ar_update_ordering/buf_reg_reg_buf

Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg_ar_update_ordering.pat

Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg_ar_update_ordering.txt (0 tests)

Random input: parallel_float/mpyf3.addf3_ar_update_ordering/buf_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

r4: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r5: a 32-bit integer register (value of st)

ldiu r1, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyf3 *ar4++(1), r3, r0

|| addf3 r4, *ar4--(1), r2 ← instruction under test

ldiu r0, r3

ldiu r2, r4

ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3/buf_buf_reg_reg
Pattern: patterns/par_src_float_buf_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.pat
↳ float_reg.pat
Manually selected input: inputs/par_src_float_buf_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)
↳ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.subf3/buf_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
ar4: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 *ar2, *ar4, r0
|| subf3 r3, r4, r2 ← instruction under test
ldfu r0, r3
ldfu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3_ar_update_ordering/buf_buf_reg_reg
Pattern: patterns/par_src_float_buf_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg_ar_update_ordering.pat
↳ float_reg_ar_update_ordering.pat
Manually selected input: inputs/par_src_float_buf_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg_ar_update_ordering.txt (0 tests)
↳ float_reg_dst_float_reg_ar_update_ordering.txt
Random input: parallel_float/mpyf3.subf3_ar_update_ordering/buf_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
ldiu ar2, ar4
ldiu ar2, ar5
mpyf3 *ar4++(1), *ar4--(1), r0
|| subf3 r3, r4, r2 ← instruction under test
ldiu r0, r3
ldiu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3/buf_reg_reg_reg
Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)
↪ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.subf3/buf_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 40-bit floating point register
---- OUTPUTS ----
r4: a 40-bit floating point register
r5: a 40-bit floating point register
r6: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 *ar2, r3, r0
|| subf3 r4, r5, r2 ← instruction under test
ldfu r0, r4
ldfu r2, r5
ldiu st, r6

Subdirectory: parallel_float/mpyf3.subf3/reg_buf_reg_reg
Pattern: patterns/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_reg_src_float_buf_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)
↪ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.subf3/reg_buf_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 40-bit floating point register
r5: a 40-bit floating point register
---- OUTPUTS ----
r4: a 40-bit floating point register
r5: a 40-bit floating point register
r6: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 r3, *ar2, r0
|| subf3 r4, r5, r2 ← instruction under test
ldfu r0, r4
ldfu r2, r5
ldiu st, r6

Subdirectory: parallel_float/mpyf3.subf3/reg_reg_reg_reg
Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_src_float_reg_dst_float_reg.txt (0 tests)
↪ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.subf3/reg_reg_reg_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
r4: a 40-bit floating point register
---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 r3, r4, r0
|| subf3 r3, r4, r2 ← instruction under test
ldfu r0, r3
ldfu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3/buf_reg_buf_reg
Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.pat
↪ float_reg.pat
Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.txt (0 tests)
↪ float_reg_dst_float_reg.txt
Random input: parallel_float/mpyf3.subf3/buf_reg_buf_reg/random.txt (100 tests)
Assembly pattern under test:
---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
ar4: an auxiliary register pointing to an array of 1 32-bit floating point values
r4: a 40-bit floating point register
---- OUTPUTS ----
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
mpyf3 *ar2, r3, r0
|| subf3 *ar4, r4, r2 ← instruction under test
ldfu r0, r3
ldfu r2, r4
ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3.ar_update_ordering/buf_reg.buf_reg
Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.ar_update_ordering.pat
Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.ar_update_ordering.txt (0 tests)
Random input: parallel_float/mpyf3.subf3.ar_update_ordering/buf_reg.buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
 ---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
 ldiu r1, st
 ldiu ar2, ar4
 ldiu ar2, ar5
 mpyf3 *ar4++(1), r3, r0
 || subf3 *ar4--(1), r4, r2 ← instruction under test
 ldiu r0, r3
 ldiu r2, r4
 ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3/reg_reg.buf_reg
Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.pat
Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_buf_src_float_reg_dst_float_reg.txt (0 tests)
Random input: parallel_float/mpyf3.subf3/reg_reg.buf_reg/random.txt (100 tests)
Assembly pattern under test:
 ---- INPUTS ----
r1: a 32-bit integer register (value for st)
r3: a 40-bit floating point register
r4: a 40-bit floating point register
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r5: a 40-bit floating point register
 ---- OUTPUTS ----
r4: a 40-bit floating point register
r5: a 40-bit floating point register
r6: a 32-bit integer register (value of st)
 ldiu r1, st
 mpyf3 r3, r4, r0
 || subf3 *ar2, r5, r2 ← instruction under test
 ldfu r0, r4
 ldfu r2, r5
 ldiu st, r6

Subdirectory: parallel_float/mpyf3.subf3/reg_reg.buf.buf

Pattern: patterns/par_src.float_reg_src.float_reg_dst.float_reg_src.float.buf_src.float.buf_dst.
↳ float_reg.pat

Manually selected input: inputs/par_src.float_reg_src.float_reg_dst.float_reg_src.float.buf_src.
↳ float.buf_dst.float_reg.txt (0 tests)

Random input: parallel_float/mpyf3.subf3/reg_reg.buf.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

r3: a 40-bit floating point register

r4: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r5: a 32-bit integer register (value of st)

ldiu r1, st

mpyf3 r3, r4, r0

|| subf3 *ar2, *ar4, r2 ← instruction under test

ldfu r0, r3

ldfu r2, r4

ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3_ar_update_ordering/reg_reg.buf.buf

Pattern: patterns/par_src.float_reg_src.float_reg_dst.float_reg_src.float.buf_src.float.buf_dst.
↳ float_reg.ar_update_ordering.pat

Manually selected input: inputs/par_src.float_reg_src.float_reg_dst.float_reg_src.float.buf_src.
↳ float.buf_dst.float_reg.ar_update_ordering.txt (0 tests)

Random input: parallel_float/mpyf3.subf3_ar_update_ordering/reg_reg.buf.buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

r4: a 40-bit floating point register

---- OUTPUTS ----

ar4: an auxiliary register

ar5: an auxiliary register

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r5: a 32-bit integer register (value of st)

ldiu r1, st

ldiu ar2, ar4

ldiu ar2, ar5

mpyf3 r3, r4, r0

|| subf3 *ar4++(1), *ar4--(1), r2 ← instruction under test

ldiu r0, r3

ldiu r2, r4

ldiu st, r5

Subdirectory: parallel_float/mpyf3.subf3/reg_reg_reg_buf

Pattern: patterns/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/par_src_float_reg_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg.txt (0 tests)

Random input: parallel_float/mpyf3.subf3/reg_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r5: a 40-bit floating point register

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r6: a 32-bit integer register (value of st)

ldiu r1, st

mpyf3 r3, r4, r0

|| subf3 r5, *ar2, r2 ← instruction under test

ldfu r0, r3

ldfu r2, r4

ldiu st, r6

Subdirectory: parallel_float/mpyf3.subf3/buf_reg_reg_buf

Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg.pat

Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg.txt (0 tests)

Random input: parallel_float/mpyf3.subf3/buf_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

---- INPUTS ----

r1: a 32-bit integer register (value for st)

ar2: an auxiliary register pointing to an array of 1 32-bit floating point values

r3: a 40-bit floating point register

r4: a 40-bit floating point register

ar4: an auxiliary register pointing to an array of 1 32-bit floating point values

---- OUTPUTS ----

r3: a 40-bit floating point register

r4: a 40-bit floating point register

r5: a 32-bit integer register (value of st)

ldiu r1, st

mpyf3 *ar2, r3, r0

|| subf3 r4, *ar4, r2 ← instruction under test

ldfu r0, r3

ldfu r2, r4

ldiu st, r5

Subdirectory: parallel_float/mpyf3_subf3_ar_update_ordering/buf_reg_reg_buf

Pattern: patterns/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg_ar_update_ordering.pat
↳ float_reg_ar_update_ordering.pat

Manually selected input: inputs/par_src_float_buf_src_float_reg_dst_float_reg_src_float_reg_src_float_buf_dst_float_reg_ar_update_ordering.txt (0 tests)
↳ float_buf_dst_float_reg_ar_update_ordering.txt (0 tests)

Random input: parallel_float/mpyf3_subf3_ar_update_ordering/buf_reg_reg_buf/random.txt (100 tests)

Assembly pattern under test:

```

---- INPUTS ----
r1: a 32-bit integer register (value for st)
ar2: an auxiliary register pointing to an array of 1 32-bit floating point values
r3: a 40-bit floating point register
r4: a 40-bit floating point register
---- OUTPUTS ----
ar4: an auxiliary register
ar5: an auxiliary register
r3: a 40-bit floating point register
r4: a 40-bit floating point register
r5: a 32-bit integer register (value of st)
ldiu r1, st
ldiu ar2, ar4
ldiu ar2, ar5
mpyf3 *ar4++(1), r3, r0
|| subf3 r4, *ar4--(1), r2 ← instruction under test
ldiu r0, r3
ldiu r2, r4
ldiu st, r5

```