CS/ECE552 Spring 2019 Extra Credit Report

1 Components Attempted

1.1 LRU

Instead of using the standard `victimway` strategy for cache eviction, we instantiated at `256 bit` register for each memory system module where a set bit in the register corresponds to the least recently used cache for that particular line. In the case that a cache eviction needs to take place (ie: both lines missing), the bit is checked to determine which cache should evict.

Tests were written to gauge the performance, and the improvement was only about 10% in the best case, and `cache_thrash` was noticably worse. But it was implemented for both `IMEM` and `DMEM`, identically for both.

Test	**Vway**	**LRU**
`cache_thrash.asm`	SUCCESS CPI:2.7 CYCLES:10351 ICOUNT:3803 IHITRATE: 99.9 DHITRATE: 33.1	SUCCESS CPI:2.9 CYCLES:10951 ICOUNT:3803 IHITRATE: 99.9 DHITRATE: 24.8
`rand_final/t_103_ctrl.asm`	SUCCESS CPI:3.7 CYCLES:1980 ICOUNT:540 IHITRATE: 70.7 DHITRATE: 68.9	SUCCESS CPI:3.7 CYCLES:1974 ICOUNT:540 IHITRATE: 70.7 DHITRATE: 71.1
`rand_final/t_107_all.asm`	SUCCESS CPI:3.6 CYCLES:1494 ICOUNT:415 IHITRATE: 69.9 DHITRATE: 65.6	SUCCESS CPI:3.6 CYCLES:1496 ICOUNT:415 IHITRATE: 69.9 DHITRATE: 65.6
`rand_final/t_111_mem.asm`	SUCCESS CPI:3.9 CYCLES:2675 ICOUNT:681 IHITRATE: 75.5 DHITRATE: 91.7	SUCCESS CPI:3.9 CYCLES:2672 ICOUNT:681 IHITRATE: 75.5 DHITRATE: 91.7

Test	Vway	LRU
`rand_final/t_112_ctrl.asm`	SUCCESS CPI:3.6 CYCLES:1898 ICOUNT:534 IHITRATE: 70.6 DHITRATE: 61.0	SUCCESS CPI:3.6 CYCLES:1894 ICOUNT:533 IHITRATE: 70.6 DHITRATE: 63.4
`rand_final/t_117_all.asm`	SUCCESS CPI:3.6 CYCLES:1640 ICOUNT:457 IHITRATE: 71.6 DHITRATE: 63.8	SUCCESS CPI:3.6 CYCLES:1630 ICOUNT:456 IHITRATE: 71.5 DHITRATE: 68.1
`rand_final/t_118_all.asm`	SUCCESS CPI:3.6 CYCLES:1801 ICOUNT:507 IHITRATE: 71.0 DHITRATE: 70.2	SUCCESS CPI:3.6 CYCLES:1797 ICOUNT:506 IHITRATE: 71.0 DHITRATE: 72.3
`rand_final/t_125_all.asm`	SUCCESS CPI:3.7 CYCLES:1839 ICOUNT:496 IHITRATE: 70.4 DHITRATE: 55.2	SUCCESS CPI:3.7 CYCLES:1827 ICOUNT:495 IHITRATE: 70.3 DHITRATE: 58.6
`rand_final/t_129_all.asm`	SUCCESS CPI:3.6 CYCLES:1559 ICOUNT:428 IHITRATE: 70.9 DHITRATE: 70.8	SUCCESS CPI:3.6 CYCLES:1555 ICOUNT:427 IHITRATE: 70.9 DHITRATE: 72.9
`rand_final/t_144_all.asm`	SUCCESS CPI:3.6 CYCLES:1578 ICOUNT:433 IHITRATE: 69.8 DHITRATE: 60.0	SUCCESS CPI:3.6 CYCLES:1574 ICOUNT:432 IHITRATE: 69.7 DHITRATE: 63.3
`rand_final/t_148_all.asm`	SUCCESS CPI:3.5 CYCLES:1595 ICOUNT:450 IHITRATE: 71.3 DHITRATE: 70.7	SUCCESS CPI:3.5 CYCLES:1589 ICOUNT:449 IHITRATE: 71.2 DHITRATE: 73.2
`rand_complex/t_5_all.asm`	SUCCESS CPI:3.4 CYCLES:1827 ICOUNT:533 IHITRATE: 71.8 DHITRATE: 59.1	SUCCESS CPI:3.4 CYCLES:1821 ICOUNT:532 IHITRATE: 71.8 DHITRATE: 61.4

Test	Vway	LRU
`rand_complex/t_6_all.asm`	SUCCESS CPI:3.2 CYCLES:2080 ICOUNT:643 IHITRATE: 72.6 DHITRATE: 64.	SUCCESS CPI:3.2 CYCLES:2086 ICOUNT:643 IHITRATE: 72.6 DHITRATE: 63.0

`lru.addr`:

```
0 1 2048 0
0 1 0 0
0 1 4096 0
0 1 0 0
0 1 6144 0
0 1 0 0
0 1 8192 0
0 1 0 0
0 1 10240 0
0 1 0 0
0 1 12288 0
0 1 0 0
0 1 14336 0
0 1 0 0
0 1 16384 0
0 1 0 0
0 1 18432 0
0 1 0 0
0 1 20480 0
0 1 0 0
```

`lru.asm`:

```
lbi r0, 0
lbi r1, 8
slbi r1, 0
add r2, r1, r1
add r3, r2, r1
add r4, r3, r1
add r5, r4, r1
add r6, r5, r1
ld r7, r1, 0
ld r7, r0, 0
ld r7, r2, 0
ld r7, r0, 0
ld r7, r3, 0
ld r7, r0, 0
ld r7, r4, 0
ld r7, r0, 0
ld r7, r5, 0
ld r7, r0, 0
ld r7, r6, 0
ld r7, r0, 0
halt
```

`lru_result.log`:

```
On lru.asm

using victimway, data hit rate is 33.3
using lru,       data hit rate is 41.7

On lru_mult.asm

using victimway, data hit rate is 12.9
using lru,       data hit rate is 22.7
```

1.2 Replace RCA with CLA

An CLA was implemented and behaved the same as our RCA over all tests. In the process of synthesis, we noticed that this reduced our critical path greatly, but not to the extent that was necessary to get our design within constraints.

We did not test any further than just testing with instructor-provided tests and tests for HW3[1].

1.3 Exception Handling

SIIC
This is treated as a particular branch instruction, still decided in decode, but branched to a constant address of the trap handler. It is handled as a `JAL` , but where the current (already incremented) PC is written to a special `EPC` register instead of `R7` .
RTI
This is the same as a standard `ret` instruction, but where the jump target is given by the `EPC` register instead of `R7` .
We did not test further than the instructor-provided tests.

1.4 Synthesis

We attempted but could not successfully meet timing paramters. We have a critical path which flows from the stall signal of the data memory to the write enable on the IDEX pipeline register, which synopsys could not optimize down. Various ungroupings, regroupings, flattenings, etc. did not managed to make this possible.

> **1. Components Attempted**

Author: Team 3: Jeremy Intan, Chetankumar Mistry, Nicholas Sielicki

Created: 2019-05-05 Sun 21:59

Emacs (http://www.gnu.org/software/emacs/) 27.0.50 (Org-mode (http://orgmode.org) 9.2.3)