MIPS Simulator: Data Cache Study + Victim Buffer
Project 5ec [EXTRA CREDIT] - CS 3339 – Spring 2016

Due: Wednesday, 4/20/16 @ noon
50 points

This assignment is entirely optional. It is composed of two independent parts, for a maximum of 50 additional project points added to your project category total. (Your project category score can overflow and help drag up your scores in other score categories in your final course average!)

You are welcome to complete only the first part, only the second part, both parts, or none of them.

PART A: DATA CACHE STUDY
[25 points]

In this part, you will use your Project 5 data cache model to study several possible cache configurations for a 1 KiB data cache and observe the relationships between block size, associativity, hit ratio, and performance.

You will vary the values of block size, number of sets, number of ways, cycles of hit latency, cycles of read latency, and cycles of write latency. You may make any changes necessary to your Project 5 data cache model to achieve this. Note that if you designed your Project 5 code well (making use of all the #define statements in CacheStats.h), no changes should be required.

The Makefile I gave you for Project 5 already understands how to set all of the #defines in CacheStats.h from the make command line. To compile your data cache model as a 4-way set associative cache with a block size of 16 bytes, 32 sets, a hit latency of 1 cycle, a read latency of 31 cycles, and a write latency of 11 cycles, you would compile your simulator with the command:

```
$ make clean; make BLOCKSIZE=16 SETS=32 WAYS=4 HIT_LATENCY=1 READ_LATENCY=31 WRITE_LATENCY=11
```

You must always run “make clean” before re-making the simulator with a different cache configuration. The configuration above results in a data cache size of 2 KiB (2048 bytes).

For this project, you will search a space of possible configurations for a 1 KiB (1024 byte) data cache. Re-compile your simulator and then collect the cache hit ratio and performance (CPI) on sssp.mips for all possible combinations of the following:
- Direct mapped, 2-way set associative (SA), 4-way SA, 8-way SA, fully associative
- Block sizes of 8 bytes, 16 bytes, 32 bytes, 64 bytes, and 128 bytes

You should end up with 24 different configurations. Make sure you choose the number of sets to always result in a 1 KiB cache.

The following tables state the latencies that you should use in each case. (Like in Project 5, the read and write latencies are different. This suggests that the next level of the memory hierarchy
Writeback dirty data to the next level of the memory hierarchy.

Replaced into the cache of whether the block is a victim.

Your victim buffer should cache the victim block and the cache block are swapped.

Write a report (which does not need to be longer than a page, or two at most!) with your results, displayed both in table format and graphically. You may choose to graph the results using any plot format you’d like. Choose one that you think best supports your written discussion. Write a few paragraphs discussing your results and conclusions. Be sure to address the following:

- Which configuration results in the best hit ratio?
- Which configuration results in the best performance?
- Why does the best hit ratio not yield the best performance?
- Why might increasing associativity result in increased hit time? What is the impact of this tradeoff on performance?
- Why might increasing block size result in increased miss latency? What is the impact of this tradeoff on performance?
- If you were choosing the best cache configuration for sssp.mips, which would you choose?
- These results are based on only one program executable. What (if anything) can you infer about what’s likely to be the best configuration for a broader set of codes?

Submit your report to TRACS as a PDF named project5sec_partA.pdf. There is no code submission for this part of the project.

PART B: VICTIM BUFFER
[25 points]

In this part, you will further enhance your simulator to add a victim buffer to the data cache. You should begin with a copy of your Project 5 submission. There is no additional code provided for this assignment.

A victim cache is a small, fully-associative cache that sits between a cache and its refill path and contains blocks that are discarded from the cache because of a miss (i.e., “victims”) that mapped to the same index. The victim cache is checked on a cache miss to see if it has the desired data before a fill request is sent to the next lower-level of memory. If the block is found in the victim cache, the victim block and the cache block are swapped. A victim buffer is a single-entry victim cache.

Your victim buffer should cache that most recent block that was evicted from the cache, regardless of whether the block is dirty or not. On a hit to the victim buffer, swap the victim buffer entry back into the cache using the same round-robin way replacement you use to decide which block to replace on a fill (and swap the current occupant of that set/way to the victim buffer). Be sure to writeback dirty data to the next level of the memory hierarchy when a victim buffer entry is evicted.
Add the following to the block of #defines at the top of CacheStats.h:

```
#ifdef VICTIM_BUFFER
#define VB_HIT_LATENCY 1
#endif
```

All victim buffer code should be surrounded by `#ifdef VICTIM_BUFFER/#else/#endif` pragmas, such that the victim buffer only exists when the code is compiled with that define. If an access hits in the victim buffer, it incurs a hit penalty of HIT_LATENCY (default: 0) + VB_HIT_LATENCY (default: 1) cycles. A normal cache hit still incurs the usual penalty of HIT_LATENCY (i.e., 0) cycles.

In addition to the Project 5 statistics, your cache model should also report:

- The number of **victim buffer hits**

Using the Project 5 Makefile, this command will build the simulator with the victim buffer enabled:

```
$ make clean; make VICTIM_BUFFER=1
```

The following is the expected result for sssp.mips. Your output must match this format verbatim.

```
CS 3339 MIPS Simulator
Cache Config: 1024 B (32 bytes/block, 8 sets, 4 ways)
  Latencies: Hit = 0 cycles, Read = 30 cycles, Write = 10 cycles
Running: sssp.mips

  7 1

Program finished at pc = 0x400440  (449513 instructions executed)

Cycles: 1365938
CPI: 3.0

Bubbles: 481710
Flushes: 51990
Stalls: 382718

Accesses: 197484
  Reads: 146709
  Writes: 50775
Misses: 11206
  Read misses: 7949
  Write misses: 3257
Writebacks: 4574
Victim buffer hits: 838
Hit Ratio: 94.3%
```

Compare this result to your Project 5 result. How much does a single-entry victim buffer improve the hit rate? How big is the performance increase?

Submit to TRACS all of the code necessary to compile your simulator (all of the .cpp and .h files as well as the Makefile) as a compressed tarball named `project5ec_partB.tgz`. 