Computer Science 146/246
Homework #3

Due 11:59 P.M. Oct 22nd, 2019

Suggested Readings Computer Architecture: A Quantitative Approach, *Fifth* Edition, Appendix B and Chapter 2
or *Fourth* Edition, Appendix C and Chapter 5

1 Pin Assignment: Victim Cache [100 Points]

1.1 Summary

This homework will help you understand the memory hierarchy and how to model caches using simulators. In this homework, we will use a Pin-based cache simulator that models a complete cache memory system (both the I- and D-cache are modeled, including L1 and L2 levels). Your job is to implement the victim cache idea introduced by Norman Jouppi from DEC in 1990 [1] (donwload here). Ideas from this paper, like victim cache and stream buffers, have been widely used in computer systems. For example, a victim cache was used in the HP PA7100 microprocessor as well as the AMD Athlon which has 8-entry victim buffers.

1.2 Inclusion Property [10 Points]

Read the paper and answer the following questions:
In the second paragraph of Secion 3.5, the paper mentions that victim caches violate inclusion properties.

A. Explain the concept of cache inclusion property.

B. The paper also mentions that the line size of the second-level cache is 8 to 10 times larger than the first-level cache line size, which also violates the inclusion property. Explain why this is true with a simple example.

1.3 CACTI [10 Points]

The paper mentions several times that the access latency of direct-mapped caches is much lower than set-associative caches. Unfortunately, the paper does not provide any evidence for this claim. In order to quantitatively compare cache access time, chip area, and power dissipation, Jouppi developed a modeling tool called CACTI. This tool was originally developed in the early 90's and is still actively developed and used today.

You can find more information about CACTI at http://www.hpl.hp.com/research/cacti/.

We have installed CACTI in cs246 for you and can be executed using the command "cacti -infile cache.cfg". The hw3/cache.cfg file contains the parameters of the cache that will be simulated. The output of the command returns multiple features of the simulated cache related to cache delays and power consumption.

Set cache size to 8KB and line size to 32B, sweep associativity from 1 to 4, set the number of banks to 1 and use 40nm technology, report the *"Access Time"* from CACTI. Create a plot with the Access Time in function of the cache Associativity. Comment on the results and attach your answer to the deliverable.

1.4 Stream Buffers [10 Points]

The paper mentions that stream buffers are only effective for unit-stride reference streams that at most skip every other or every third word.

A. Explain how a stream buffer works with a reference stream that skips to every third word.

B. How would you design stream buffers differently to handle non-unit-stride?

1.5 Victim Cache Implementation [70 Points]

Implement a victim cache for the L1 Dcache as presented in Jouppi's paper. Read the code for the cache simulator carefully and make sure you understand how it works before you start.

1.5.1 Cache Configuration

Currently the cache simulator is fully configurable for block size, cache size and associativity. We have preconfigured the cache simulator to read in cache configuration input from a configuration file that you can specify in the command line using the "-config" flag. An example cache configuration file is included called "config-base". The format of the configuration file is :

```
l2_block_size, l2_cache_size, l2_cache_associativity
l1_icache_block_size, l1_icache_size, l1_icache_associativity
l1_dcache_block_size, l1_dcache_size, l1_dcache_associativity
```

1.5.2 Using Knobs to Specify Configurations

To run a simulation on Unix utility ls, using the included configuration file "config-base":

```
$ pin -t obj-intel64/hw2.so -config config-base -- /bin/ls
```

This will generate the file "cache.out" which has statistics for the program execution.

You can also specify the name of the output file using knobs:

```
$ pin -t obj-intel64/hw2.so -config config-base -outfile base.out
-- /bin/ls
```

This will instead generate the base.out file instead of the cache.out file.

You can also specify the number of instructions that you want to simulate using the "-max_inst" knob:

```
$ pin -t obj-intel64/hw2.so -c config-base -outfile base.out
-max_inst 1000 -- /bin/ls
```

Once you see your Pin tool working correctly, you can remove the "-max_inst" option in your command line. By default the program will run for 1 billion instructions, which takes a couple minutes for each benchmark.

1.6 Benchmarks

Please present your results for the following two benchmarks: *libquantum* and *hmmer*. The programs are run natively as follows:

```
$ /home/cs246/benchmarks/libquantum_O3 400 25
$ /home/cs246/benchmarks/hmmer_O3 /home/cs246/benchmarks/inputs/nph3.hmm
/home/cs246/benchmarks/inputs/swiss41
```

Notice that *hmmer* takes two files: *nph3.hmm* and *swiss41* as inputs. Pass the above application commands to the Pin tool in place of "/bin/ls". Please run the benchmarks from your pin tool directory and keep the full path to the input files.

1.7 Evaluation

In this study, we assume that all cache parameters are fixed as in the default "config-base" file except the associtivity of L1 Data Cache.

A. Sweep the associtivity of L1 Data Cache from 1 (direct-mapped), 2, 4, until 8. You can find the total misses of D-Cache from the output file ("cache.out" by default). Plot the misses for both benchmarks while sweeping the associtivity. Run both benchmarks with perf, and highlight the native total misses in the plot for the associativity the caches of the cores in the server. Explain your findings.

B. Implement the victim cache for L1 Data cache. You need to parameterize the number of entries in the victim cache.

C. Sweep the number of entries in the victim cache from 1 to 8, increasing 1 each step, with direct-mapped L1 Data Cache. Plot the misses for both benchmarks while sweeping the number of entries of victim cache. Explain your findings.

D. Jouppi claims that victim cache benefits would be signifcantly smaller in associative caches. Simulate an 8-way associative L1D with a victim cache and compare the benefits with the direct-mapped case.

2 Submission Instruction

Please combine your answers to the inclusion and CACTI questions with the miss plots and your findings from your Pin assignment into a single PDF file. Please include all the code you have written for this assignment including your pin tool that simulates the victim cache.

References

[1] Norman P Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Computer Architecture (ISCA)*, 1990.

Updated October 11, 2019, Iulian V. Brumar