CS203 Fall ’17 - Lab 4

Due: Thursday, November 30 midnight (11:59:59 pm)

Team: Jingnan Cao(861308788), Yue Wang(861306322)

# Objective

The goal of Lab 4 is to design and implement a simple cache simulator, with support for direct-mapped, set-associative, and fully-associative mapping.  
If you have any questions, please post to iLearn as others may be having the same issue.

# Memory Access Traces

To facilitate the testing of your cache simulator, we will utilize two sets of traces collected from a run of gcc. The traces gcc-10K.memtrace, and gcc-1M.memtrace are available at the following git repository: <https://bitbucket.org/danwong/cs203-labs-f16> under the Cachesim subdirectory. The traces contain ~10 thousand and ~1.5 million entries. A sample of the trace is given below:

L -200 7fffe7ff088

L 0 7fffe7fefd0

S 8 12ff228

S 8 12ff208

L 0 a295e8

Each line in the trace file contains the memory instruction type (L = load, S = store), the offset, and the memory address in hexadecimal. *Note that this trace was obtained from an x86 machine, and thus the memory address is 44-bits. For the purpose of this class, we can assume 32-bits and you can truncate the most significant 12 bits.*

# Cachesim

Now that you have a better idea of how simulators are implemented, this project will be a bit more open ended. You will implement from scratch a C++/Java/Python based cache simulator (let’s call it Cachesim). Cachesim is a performance simulator, with the main focus on collecting cache miss and cache hit rates. Your simulator must meet the following requirements.

1. Command line argument for input trace file name
2. Command line argument for cache configuration
   1. The cache configuration will take several parameters including:  
      - Total cache size in bytes  
      - Cache block size in bytes  
      - Number of ways
3. Initialize all entries, tag, and LRU-bits of the cache to 0’s.
4. Output the cache miss and hit rate, as well as the number of sets, ways, and number of address bits for the tag, index, and offset.

For sample code of how to read a file, refer back to the pipesim source code.

For simplicity, we can make the following assumptions:

1. Maximum of 4MB cache size
2. Assume Least Recently Used (LRU) cache replacement policy

Please include a printout of your simulation run when executing gcc-10K.memtrace and gcc-1M.memtrace. In addition, ***include documentation on how to compile and run your code***.

1. Submit your source code and your printout to iLearn in a single zip file.
2. Submit your answers to questions to GradeScope as a pdf file.

# Questions

1. (10 points) What is the cache miss rate of the given traces and cache configuration? Assume we have a 512KB cache and 16B block size.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Trace*** | **Direct** | **2-way** | **4-way** | **8-way** | **Fully Assoc.** |
| *gcc-10K* | 7.62% | 7.56% | 7.56% | 7.56% | 7.56% |
| *gcc-1M* | 1.24% | 0.94% | 0.92% | 0.92% | 0.92% |

1. (10 points) **Effect of associativity.** Assuming a 256KB cache and 32B block size. How does increasing the number of ways affect cache miss rate? Plot the miss rate for the two traces versus the number of ways (1,2,4,8,16). What do you observe and why?

Observation:

1. Increasing associativity after 2-way does not affect the miss rate much. The reason may be that the cache size is big enough, and not many addresses hash to the same set.

2. The miss rate for gcc-1M is much lower than that of gcc-10K. This may due to that the locality is held in a long time, thus the more memory access operation we have, the lower the miss rate is.

1. (10 points) **Effect of block size**. Assuming a 256KB 2-way set associative cache. How does varying the size of the block affect cache miss rate? Plot the miss rate for the two traces versus the block size (2B, 4B, 8B, 16B, 32B, 64B). What do you observe and why?

Observation:

1. The miss rate decreases as the size of block increases. This may due to the different cost to replace the same amount of data in the cache. For a cache using block size of 2 Bytes, it will have to trigger 32 cache misses to replace 64Bytes data in the cache. For a cache using block size of 64 Bytes, one miss can replace 64 Bytes of data.

This also shows the spatial locality of the program. Since if the data used is sparsely distributed and cache size remains the same, a cache with bigger block size will only perform worse since it has less entries, and just like a cache will smaller size.

1. (10 points) **Effect of cache size**. Assuming a 2-way set associative cache and 32B block size. How does varying the size of the cache affect cache miss rate? Plot the miss rate for the two traces versus the cache size (64K, 128K, 256K, 512K, 1M, 2M). What do you observe and why?

Observation:

1. The cache miss rate decreases as the size of the cache increases. Since the bigger the cache is, the more entries the cache can hold.
2. The miss rate decreases much more slower when cache size is big. This may because that most entries has been saved in the cache, and more entries do not contribute to the performance.