**Cache Simulation Program**

**CSIS 5814**

**Spring 2020**

For this project you will create a data cache simulator. The main reason for this simulator is to use it to determine optimal parameters for a given access sequence. As in your previous project, this project has several levels of complexity. The lowest level is a simple direct mapped read-only data cache. Higher levels include set associative and fully associative caches. The rest of this discussion here will focus on the levels above the minimum.

The simulator you implement needs to work for an N-way set associative cache which can be of arbitrary size (in power of 2) up to 64KB total size. You should assume the memory is byte addressable.

The input to your simulator should be a file named ***addresses.dat.*** Your simulator will have to read a series of memory addresses from the file. Each line in the file contains a character 'r' or 'w', white spaces, and a positive decimal integer representing an address. If you are not implementing a writeable cache, your program should ignore the lines that starts with a 'w'. For each address, you should model the reading/writing a byte from the simulated cache, recording a hit or a miss. For this assignment, it does not matter what data is read or written, so you do not have to model actual memory.

Your simulator need to support the following three command line parameters (use *int main (argc, argv[])*):

|  |  |
| --- | --- |
| -s | Cache Size (in Bytes) |
| -a | Associativity (not needed for direct mapped) |
| -b | Block size (in Bytes) |

All three values should be integer powers of 2. All sizes are in Bytes.

A version of the [addresses.dat](http://kaschueller.people.ysu.edu/classes/s2020/5814/Cache/addresses.dat) is available. For Set-Associative and Fully-Associative models, use the Round Robin replacement policy for choosing the cache block to be replaced in the set, If you are implementing a writable cache, always choose a clean block before choosing a dirty block. Assume a Write Allocate policy when writing to a block that is not in cache.

I have designed three classes that your program **must** be based on. They are:

|  |  |  |
| --- | --- | --- |
| a single cache entry | [CacheEntry.h](http://kaschueller.people.ysu.edu/classes/s2020/5814/Cache/CacheEntry.h) | [CacheEntry.cc (complete)](http://kaschueller.people.ysu.edu/classes/s2020/5814/Cache/CacheEntry.cc) |
| a cache set | [CacheSet.h](http://kaschueller.people.ysu.edu/classes/s2020/5814/Cache/CacheSet.h) | [CacheSet.cc (partial)](http://kaschueller.people.ysu.edu/classes/s2020/5814/Cache/CacheSet.cc) |
| a cache | [Cache.h](http://kaschueller.people.ysu.edu/classes/s2020/5814/Cache/Cache.h) (coming) | [Cache.cc (partial)](http://kaschueller.people.ysu.edu/classes/s2020/5814/Cache/Cache.cc) (coming) |

Your simulator will have to read a series of memory addresses from the file *addreses.dat* which

Your program **must be able to run on the departmental server** (gemini.csis.ysu.edu).

**Program output**

The output of your simulator should look like:

>cache-sim -s 1024 -b 32 -a 2

cache size: 1024 bytes

block size: 32 bytes

associativity: 2

total loads: (depends on data)

total writes: (depends on data/level of project)

cache hits: (depends on data)

cache misses: (depends on data)

miss rate: (depends on data)

The maximum values your simulator needs to support is 65536 for the cache size, 256 bytes for the block size, and full associativity (size/block size).

**Report:**

In addition to your program, you must turn in a written report. Your report must include three line graphs. The Y-axis will be the miss rate. The X-axis will vary as follows:

1. The X-axis will be the size of the cache (given a fixed block size of 32 bytes and fixed associativity of 4). Your cache sizes should be 1KB, 2KB, 4KB, 8KB, 16KB, 32KB, and 64KB.
2. The X-axis will be the block size of the cache (given a fixed cache size of 8192 bytes and fixed associativity of 4). Your cache block sizes should be 1, 2, 4, 8, 16, 32, and 64 bytes.
3. The X-axis will be the associativity of the cache (given a fixed cache size of 8192 bytes and a fixed block size of 32 bytes). Your associativity should be 1, 2, 4, 8, 16, 32, and fully associative.

**Grading**

This project has 4 levels of complexity:

| Level | Mapping | Read/Write | Replacement Policy | Write Policy |
| --- | --- | --- | --- | --- |
| 0 | Direct | Read only | NA | NA |
| 1 | Set Associative | Read only | LRU | NA |
| 2 | Set Associative | Read and Write | LRU | Write through |

Completion of level 0 of this project will result in an 80% on project grade for the course. By completing additional levels, the project grade can be affected as follows:

|  |  |  |
| --- | --- | --- |
| Level completed | Max grade on project | |
| Undergraduate | Graduate and Honors |
| level 0 | 70% | 60% |
| level 1 | 90% | 80% |
| level 2 | 115% | 100% |