Program #3

Due Friday, Dec. 7, 2018 at 4:59pm

Collaboration

You must work on this assignment by yourself. You may not look up solutions for any part of this project from any resources (e.g. previous years' solutions, the Internet, other textbooks, etc.) including code you have written for similar assignments in other courses. You are allowed to look up C++ standard library information, and you are allowed to look online for explanations of compiler errors. You are allowed to discuss this project with the instructor, and peer tutors. You are not allowed to discuss the program with any other students or non-students until after the submission due date and time.

Simulating Cache Behavior

In this programming assignment, you will create a cache simulator which will be used to simulate different cache configurations for different access patterns.

Specifically, you will need to create a `Cache` class which is configurable with respect to capacity, block size, and set associativity. These parameters will be specified in a configuration file so as to not require recompilation of your code.

Code Details

You will need to copy the code at `~kshaw/share/cs301/prog3`. In this directory, you will find:

- `Makefile` – makefile for compiling your `Cache` class
- `driver.cpp` – code that reads configuration and input file, passing values to a cache
- `config-*.txt` – two example configuration files (direct mapped and 4-way set associative)
- `in*.txt` - example input files
- `out*.txt` - example output files
- `Cache.h` - a header file for the `Cache` class you are implementing. Remember that caches are basically arrays of sets and sets are arrays of blocks.

You are not allowed to add any methods or data fields to the public interface for the `Cache` class.

There are 3 methods that print out values, namely `initialize`, `printStatistics`, and `printContents`. The format for `initialize`'s output can be found in the output files. The following should be the format for the remaining two:

`printStatistics`:

ACCESSES 3
HITS 2
MISSES 1
HIT RATE 0.666667

`printContents`:

```
****** SET 0******
Index 0: tag 0 valid 1 lru 3
Index 1: tag 2 valid 1 lru 2
```

```
Index 2: tag 8 valid 1 lru 0
Index 3: tag a valid 1 lru 1
*****************
****** SET 1******
Index 0: tag 1 valid 1 lru 1
Index 1: tag 9 valid 1 lru 0
Index 2: tag 0 valid 0 lru -1
Index 3: tag 0 valid 0 lru -1
*****************
```

The lru field is -1 when the block is invalid. In this scenario of 4-way set associativity, a lru value of 3 would indicate that that block should be the next to be evicted from the set.

(Please note that you should do error checking on the parameters passed in to configure the `Cache`. You should make sure everything is a power of two AND that block size times associativity divides equally into capacity.)

Deliverables and grading

- This programming assignment will be worth half the number of points as the first two programming assignments.
- Your grade will be based 65% on correctness, 10% on design, 10% on your design document, 15% on style (including commenting).
- You must use the makefile I provided. **Your output must match the format I have specified in the example output files or you will lose points for correctness.** You should not have extraneous output.
- Your design document (named DESIGN) should explain at a high level your design. You should describe all classes you created and how they interact with one another, including a discussion of important methods. You should also describe all data fields and data structures you have added into your classes and why.

Regarding testing

I am expecting you will test your code sufficiently, but I am not asking you to submit test files. Here, however, are some suggestions for how to systematically make sure your code is working. Create a set of relatively short input files (~25-30 accesses) and configuration files for testing. You should construct these input files and configuration files so that they demonstrate that your cache is behaving correctly and that changing the cache's configuration results in different outcomes (hit rates). Some specific things you'll want to make sure you test.

- Demonstrate the use of multiple cache sets. (You should show this for both direct-mapped and set-associative caches).
- Demonstrate that when cache conflicts exist in a set, appropriate blocks are evicted. (You should show this for both direct-mapped and set-associative caches).
- You should create a set of input files and configuration files that show that your `Cache` works (and produces different outcomes) when cache capacity, block size, and associativity are varied. To do this correctly, you should have at least two configurations where capacity differs, but the other two parameters are the same. You would then choose an input file and run your cache on it for these two different configurations. The input file you choose must result in your seeing different outcomes for the two configurations.

  You would then perform the same set of experiments for the other two parameters. Meaning, you would then have at least two configurations where block size differs, but the other two parameters are the same. You would choose a single input file to run on

your cache with the two configurations; the input file you choose must result in you seeing different outcomes for these two configurations. Then, you would have to have at least two configurations where associativity varies, but the other two parameters are the same. You would choose a single input to run on your cache with the two configurations; the input file you choose must result in your seeing different outcomes for these two configurations.

(You will likely have to construct different input files in order to exercise the different aspects of the cache configuration.)

Submitting Your Work

You must submit your work using a script called turnin **by Friday, Dec. 7 at 4:59pm**.

Remember to comment your code, to put your name in each file, and make sure you deallocate any dynamically allocated memory. Follow the course code guidelines.

While you're in your solution directory, type the following command:

```
turnin -v -c cs301 -p prog3 *
```

Only submit from in your program 3 directory.