Assignment 4 — Program 4: Cache Simulator

Dr. William Kreahling

April, 2014

Overview

You will be reading from standard input and writing to standard output. If you use input redirection from Linux you can run the program like so:
`java myProgram < inputFile`

Your assignment is to write a program that simulates the behavior of a cache. The program will read data from *standard input* and produce statistics about a cache 'trace' to standard output. The input will contain cache configuration information followed by memory references consisting of hexadecimal addresses, one per line. The first three lines of the input specifies the number of cache sets, the number of elements of each set (associativity level), and the size of a cache line, in bytes. Following the first three lines each line consists of 3 colon separated values representing the type of memory access (R or W), the size, in bytes, to read or write, and the memory address in hexadecimal notation. *You should check if each reference's address is properly aligned.*

I have provided example input below, which is also available in the file trace.dat, containing the addresses that correspond to the example references described on page 460 and 461 of your text. *Note: the addresses in the trace.dat are hexadecimal word addresses rather than decimal byte addresses.*

```
Number of sets: 2
Set size: 2
Line size: 8
R:4:58
R:4:68
R:4:58
R:4:68
R:4:40
R:4:c
R:4:40
R:4:48
```

Details:

- The program should be written in a programming language that can be successfully compiled and executed on agora.
- You should check that the number of sets does not exceed 8K and the associativity level does not exceed 8.
- You should also check that the line size is at least 4 and that the number of sets and line size will be a power of 2.
- Your cache simulator should use an LRU replacement algorithm.
- This cache is a write-back, write allocate cache.
- NOTE: There is **no** actual data for this cache. This is a simulation. You will be given enough information to determine if a memory address would map into a cache of a given size. Then you will keep enough information to track what blocks in the cache *would* have data stored inside them. However, there is no data being provided!

- The cache simulator output should output the following information (in order):
  1. print information about the cache configuration used.
  2. print information for each reference.

     This information is: access type, the address, tag, index, offset, result (hit or miss) and number of memory references to the level of memory below the cache. Note that the address and the tag should be printed in hex. The index, offset, and memory references should be printed in decimal.

     This information will help you debug problems with your simulator. Though it is not required, you may also wish to write a function to dump the state of the cache to help debug problems.
  3. After the last reference the cache simulator should output the following summary information:
     (a) the number of hits
     (b) the number of misses
     (c) total accesses to the cache
     (d) hit ratio
     (e) miss ratio

The output below and the file trace.stats, contains the output of my cache simulator after processing the configuration information and the references in the trace file *trace.dat*. You should try to make your output match mine as closely as possible to facilitate grading of the assignment.

```
Cache Configuration

   2 2-way set associative entries
   of line size 8 bytes


Results for Each Reference

Access Address    Tag   Index Offset Result Memrefs
------ -------- ------- ----- ------ ------ -------
  read       58       5     1      0   miss       1
  read       68       6     1      0   miss       1
  read       58       5     1      0    hit       0
  read       68       6     1      0    hit       0
  read       40       4     0      0   miss       1
  read        c       0     1      4   miss       1
  read       40       4     0      0    hit       0
  read       48       4     1      0   miss       1


Simulation Summary Statistics
-----------------------------
Total hits       : 3
Total misses     : 5
Total accesses   : 8
Hit ratio        : 0.375000
Miss ratio       : 0.625000
```

- You should use good programming style, which includes descriptive variable names, abstraction, modularization, and comments.
- You should comment your program so that others (e.g. the instructor) can understand it.
- You should have comments before each function/method and each major block of code.
- You should comment every variable in your code.
- Your output should match mine *exactly*.

- You should also have comments at the top of the file indicating your name, this course, the assignment, and the command used to compile/interpret your program.

Submit:

Due: May,2 2014.
Submit via handin: **All** files needed to compile and run your program and a README.txt file explaining how to compile and run your program. Turn in all files necessary to build and execute your program.

```
handin.<course #>.<section #> <assignment #> <list of files>
```

```
handin.350.1 4 cache.c README.txt
```

Sample input/output

- fin1.dat
- fin1.stats
- assoc.dat
- assoc.stats