EE382M – 15: Assignment 0

Professor: Lizy K. John
TA: Jee Ho Ryoo
Department of Electrical and Computer Engineering
University of Texas, Austin

Due: 11:59PM September 1, 2014

1. Introduction and Goals

The goal of this assignment is to set up your account for this class, to sign up for required newsflash/scribing, and to self-assess your programming skills required for EE382M-15.

This assignment has three sections. Everyone will be required to do the first two sections while the last programming part of the assignment is for those who would like to assess their own programing skills required for this class. If you can complete the programming part in less than a week, you are likely to enjoy the programming assignments in this class. If it takes more than a week to complete this assignment, please consult the instructor.

1.1 Graded Items

The following items need to be completed by the deadline

1. Create an account on Piazza
2. Sign up for Newsflash
3. Sign up for Scribing

1.2 Sample Assignment

This is for self-assessment and nothing needs to be turned in.

1. Develop a single-core cache simulator in C

2. Required Assignment

2.1 Piazza Account

Piazza is a main communication method we will be using in class. Any general question from course logistics to assignments must be posted on Piazza. This is employed in order to reduce the amount of time spent on answering duplicate questions by the teaching staff as well as a single knowledge base for students to look for answers. The teaching staff will check the Piazza regularly to answer questions. When you ask a question, please specify which category your question belongs to. At this point, there is only one category *logistics* where you can ask any general question regarding this course. We will create separate categories such as *assignment1* as the semester goes on to post your question under a specific category.

In order to sign up, please login to Canvas course management system (https://courses.utexas.edu/). Under *Courses* tab, click *15-COMP PERF EVAL/BENCHMARKING*. Now, you should be able to see the menu on the left side.

Please click on *Piazza*. If you do not have an account, you will be asked to create an account. Once you log in, you should be able to see the first welcome post. This completes this part of the assignment.

2.2 Newsflash Sign Up

In the beginning of each class, a student will be presenting any news related to computer architecture (1-2 slides, 3 minutes). Each student has to sign up for one specific date. Please go to this link on Canvas, and select *Newsflash*. Click *Sign Up* on your preferred date, type your name and click *Sign Up.* It will be first come, first served. Therefore, if another student has already selected a date and submitted, then this date will not be able to be selected by others (the sign up link will be gone). To be fair, the link will be public on Canvas at 4:00PM CST on Aug 29, 2014.

2.2 Scribing Sign Up

In every class, one student will be scribing the lecture notes. Although the lecture note will be uploaded on the class website, it is beneficial for students to prepare for the exam as some concepts may not be explained on slides in detail. Each student has to sign up for one specific date. Please go to this link provided on Canvas, and select *Scribing*. Click *Sign Up* on your preferred date, type your name and click *Sign Up.* It will be first come, first served. Therefore, if another student has already selected a date and submitted, then this date will not be able to be selected by others (the sign up link will be gone). To be fair, the link will be public on Canvas at 4:00PM CST on Aug 29, 2014.

3. Sample Assignment

This part of the assignment is for those who would like to assess their own programing skills required for this class. If you can complete this programming part in less than a week, you are likely to enjoy the programming assignments in this class. If it takes more than a week to complete this assignment, the class assignments may be too challenging for you.

3.1 Single core cache simulator

For this part of the assignment, you are asked to create a simple, trace-driven cache simulator. The simulator should be capable of modeling a direct-mapped, set-associative or fully associative cache, with arbitrary cache sizes, block sizes, associativity and latency parameters. The simulator should be able to take command-line parameters and an address trace file name as input. It should produce a hit ratio and average memory access time (AMAT).

Your simulator must be able to take the following cache parameters as command-line inputs:

- Cache size in bytes (powers of 2) (up to 16MB)
- Block size in bytes (powers of 2)
- Set associativity (powers of 2)
- Cache hit latency in clock cycles
- Cache miss latency in clock cycles
- Address trace file-name

The cache should implement the least recently used (LRU) cache replacement policy for multi-way caches. Assume that the tag check latency occurs in parallel with the cache hit latency.

Your simulator should take the above parameters in that *exact order*. Do not vary the input format in any way. The final input parameter will be the trace file name. Here is an example of how your simulator should run:

```
% cachesim 32768 32 1 2 50 tracefile.txt
```

This run will simulate a direct-mapped 32KB cache with a 32-byte block size, 2 cycle hit latency, and 50 cycle miss latency. The trace file will contain hexadecimal addresses up to 64 bits and will be in the following format:

```
FF2000
102004
A02008
200C
2010
20FAB14
```

Your program will output a hit ratio and AMAT based on the input trace file. The output should be presented in the following format:

```
% cachesim 32768 32 1 2 50 tracefile.txt
32768 32 1 2 50
HIT RATIO: 0.953
AMAT: 4.256
%
```

Print the two lines of output and an initial line showing the configuration to the screen (stdout). Make sure to use all capital letters, put a space after the colon, and only print to three decimal places.

Make the output to be appended to a file called RESULTS. So if you run the cache simulator with 10 configurations, it will print the 10 configurations and the 10 sets of results to the same file. You can try a variety of address sequences to make sure that different block sizes and different associativies are working fine.

3.2 Assignment template

A sample input trace is provided with the assignment template. It is written in hexadecimal, so it is your job to convert it to the format you prefer. You are responsible for implementing the cachesim in a file named *cachesim.c* file in the assignment template. Your code will be compiled on gcc version 4.6.3. The assignment template is provided with a simple Makefile. Do not modify this file. You are responsible for modifying *cachesim.c* file. You can have any number of files (although one file should do) as long as they use the following C extensions: .h, .c. Comment your code.

3.3 Submission instructions

This part of the assignment is just a sample assignment and you do not have to submit anything.