Homework #5 – Cache Simulator

Due: Nov 30, 2015

Objectives: Become familiar with cache memory organization and performance. Figure out how to test a program.

Input: Your program will be invoked with several command line arguments, each of which will be an integer:

`% hw5` *<S> <E> <B> <H> <M>*

where

S = the number of sets
E = the number of lines per set
B = the block size in bytes
H = the hit time in cycles
M = the miss time in cycles

Your program will simulate the performance of a memory system with a single level cache (i.e., no L2 or L3).

For example:

`% hw5 64 8 64 4 100`

would be used for a 32 Kbyte 8-way set associate cache, where a cache hit requires 4 cycles and a miss incurs an access to main memory that takes 100 cycles.

In addition, your program will take input from **stdin**. This input will consist of a sequence of numbers. Each number will be on a separate line and will be consist of 8 hex digits.

Main memory is assumed to be byte addressable and addresses are 32 bits in length. The **stdin** input to your cache simulator is a "memory trace" reflecting the execution of some program, which we can call the "target program." The memory trace reflects every memory reference made by the target program. We do not care about the execution of the target program, or which operations it performs, or what computations and output it produces. We are only concerned with how much time it spends fetching data from memory. In fact, we will not even see the target program's code; all we have is the memory trace collected (somehow) from some execution of the target program.

We will assume the target program only reads from memory and you should ignore writes. During its execution, the target program will only read data from memory; the memory trace contains the addresses that were accessed, in order. We will not see or care about the actual data values. Each memory access is assumed to read a single byte, not multiple bytes.

Output: Your program will print the following information:

`Number of hits =` *<decimal integer>*
`Number of misses =` *<decimal integer>*
`Miss rate =` *<fraction>* **`%`**
`Total cycles =` *<decimal integer>*

Homework #5 – Cache Simulator

Your output should match this, character for character, so that we can test your program automatically. Print no tabs and use single spaces before and after the "=". Put no spaces at the end of the lines. The fraction should be printed with code like the following:

```
double d;
…
printf ("Miss rate = %4.2f %%\n", d);
```

Replacement Policy: Your simulation should implement LRU (Least Recently Used). In other words, when you must select a cache line for eviction, choose the one which has last been accessed the furthest in the past.

Handling Errors: As always, you need to think about what errors might reasonably occur and print appropriate error messages to **stderr**.

Submission: Send a single email to the grader. Include as a single attachment, the file **hw5.c** with

Subject: CS201, HW5, <your name>

Your program will be compiled and tested on the PSU Linux machines with the following command:

```
gcc -Wall -O1 –m64 hw5.c
```