Lab 5a (the first of three parts for lab 5)

Assigned: Wednesday, 8 October 2009

Due: 11:59pm, Tuesday 13 October 2009 by attachment as an email to labturnin.cs341l@gmail.com

You should attach your program with your last name and the lab number including the letter "a" to show that this is the first part in the name of the file, e.g., “crandall-lab1.S”.

No late assignments will be accepted unless circumstances warrant an extension for the whole class (e.g., server is down all weekend, etc.). **All code must be implemented in C** (no C++ or any language other than straight C). You are expected to do your own work as an individual effort, copying the C code of others is considered cheating.

The purpose of lab 5 as a whole is to write a cache simulator to demonstrate your understanding of caches, explore some of the tradeoffs of different cache designs, and work on your C/UNIX skills. Lab 5a focuses mainly on the latter.

Lab 5a is to write the command line parsing code and the code to open the trace file and read its values. The prototype for your cache simulator will be:

./lastname-lab5a -A [direct,1,2,4,8,16,full] -S **NNN** -B **NNN** tracefilename.txt

The A argument is associativity, and can be "direct" for direct mapped, "1", "2", "4", "8", or "16" for set associative, or "full" for fully associative.

The S argument is the cache size in KB, with a valid range from 1 to 8192.

The B argument is the block size in bytes, with a valid range from 1 to 1024

The final argument is the name of a file which will be a text file (the trace file) that looks like this:

```
0x7ff85467
0x08048924
...
```

Your program for 5a (which will be a skeleton of your complete lab 5) should read the command line arguments, give an error message if arguments are missing or out of range, or print a message with the arguments read and then open the file and read it line-by-line, printing the decimal equivalent of the hexadecimal value on each line until the end of the file is reached.