Project #2 (Due November 2)

NOTE: This assignment can be done by the same teams as Project 1.

The goal of this assignment is to study the effect of caches on CPU performance by simulating the instruction and data caches in the pipelined simulator, CPU.c, that you developed in Part 1 of Project 1 (with *prediction_method = 0*). Your CPU+cache simulator will stall the pipeline if, while fetching an instruction in the IF stage, the instruction is not be found in the L1 instruction cache. Similarly, the pipeline will stall if while loading or storing data in the MEM stage, a miss occurs in the L1 data cache. The simulator should accept the same command line arguments as *CPU.c* and read the following memory hierarchy configuration parameters from a configuration file "cache_config.txt":

1. the size of the L1 Instruction cache in Kilo Bytes
2. the associativity of the L1 Instruction cache
3. the L1 Instruction cache block size in bytes
4. the size of the L1 data cache in Kilo Bytes
5. the associativity of the L1 data cache
6. the L1 data cache block size in bytes
7. the memory access time in cycles (the L1 miss penalty).

All seven parameters are integers and, for simplicity, assume that each of the first six parameters is a power of 2 (see for example, the following configuration file).

A skeleton for the extension of single cycle *CPU.c* to include the effect of caches is given in *CPU+cache.c*. This skeleton, which is very rough and may not even compile, includes the file *cache.h* that contains a template for the data structures and functions that you may use to implement a cache. Your task is to complete the implementation of *CPU+cache.c* and *cache.h* assuming the following:

- All the blocks in the caches are initially invalid.
- The "write allocate" policy is applied on a write miss
- The "write back" policy applies
- LRU block replacement (implemented using an LRU stack for each cache set).

In addition to the total number of execution cycles, the simulator should report the total number of memory accesses and the miss rates for the caches. As in project 1, you should write your own simple test programs to verify the correctness of your simulation.

After completing and testing your implementation, you should use the same traces as in project 1 and conduct the following experiments, assuming a memory access time of 20 cycles:

Experiment 1: To examine the effect of the block size, simulate a system with equal data and instruction cache sizes, and consider three different sizes for each of the caches (1KB, 16KB and 128KB) with 4-way set associativity and four possible block sizes (4B, 16B, 64B, and 256B). For each of the two long traces, build a table similar to the one shown below to report the miss rates and the execution time.

	I cache miss rate			D cache miss rate			Execution time (in cycles)		
	1KB	16KB	128KB	1KB	16KB	128KB	1KB	16KB	128KB
4B									
16B									
64B									
256B									

Experiment 2: In this part, you will try to design an optimum cache system. Specifically, from experiment 1, try to determine different sizes for the I and D caches as well as different block sizes for the caches such that your system will produce good results without over provisioning your caches (without using larger than needed caches). Run your simulator with the chosen cache sizes and block sizes and report the execution time.

Experiment 3: To examine the effect of associativity, consider a 32KB I cache and a 32KB D cache each with 32B blocks. Use a bar graph to plot the miss rates for associativity = 1, 4 and 8.

What to submit (the TA may provide more specific instructions):

1) One file with your source code for the *CPU+cache.c* and *cache.h* (which includes all functions and declarations).
2) One file with
    a. The result of running your simulation with *trace_view_on* = 0 for each long trace file and the parameters specified in each experiment
    b. The tables and bar graph specified in the experiments
    c. Your remarks and explanation of the results and the conclusions that you draw from the experiments.