Computer Organization, Spring 2016

Lab 5: Cache Simulator

Due: 2016/6/20

1. Goal

Cache performance is important for system performance. In this lab, you are asked to simulate cache behaviors by C/C++ style cache simulators. By this training, you will understand the performance difference between different cache architectures.

2. Basic Problem (50%)

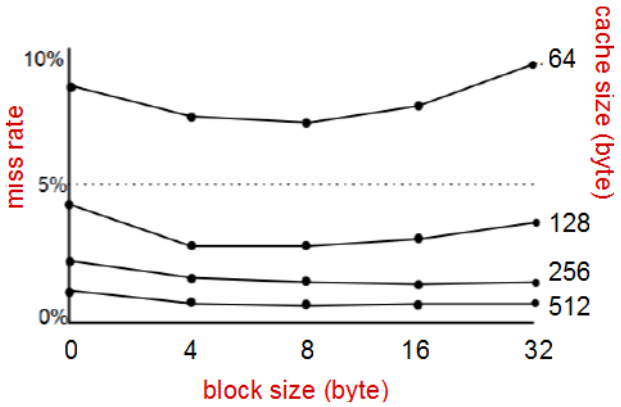
You should trace the memory addresses in either your single cycle CPU in lab 3 or pipelined CPU in lab 4. You have to implement instruction "jump" in your pipelined CPU if you would like to use the pipelined version from lab 4. The supplied files are described as follows:

a. "lab5_test_data.txt" -
The file is a 3-dimensional matrix multiplication program, which is the input of your CPU. You can find the explanation of the code in “lab5_test_data_assembly.txt”.

b. "Instr_Memory.v" -
We enlarge the instruction file from the previous lab so that it can fit the input size of "lab5_test_data.txt".

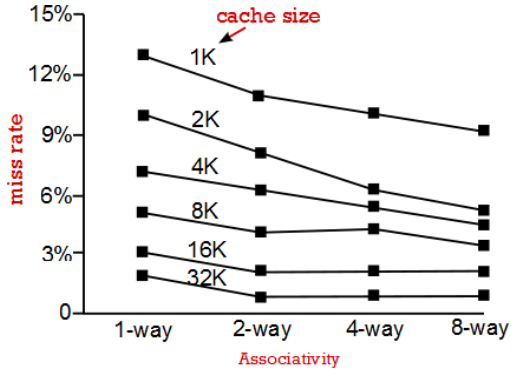
c. "TestBench.v" -
A verilog testbench, please refer to it and then modify your CPU to get the memory traces when the program is running (make sure the registers or wires in your CPU is named correctly so that it can be correctly called by TestBench.v). After running the program, you will get two output files: "ICACHE.txt" and "DCACHE.txt". These two files are the simply the memory traces of ICACHE (instruction cache) and DCACHE (data cache) respectively.

d. "direct_mapped_cache.cpp" -
A simple cache simulator. You should modify this simulator so that it can output the miss rate of the cache. **Hint: see pdf of chapter 5 in page 6.** Please take the file "ICACHE.txt" and "DCACHE.txt" that are produced by the CPU, as inputs of the simulator and then run it. First, change the parameters (cache size or block size) when you do the simulation. Then draw a graph as the following example and describe the reason of rise and fall of the lines in the report. (Please separate ICACHE from DCACHE)



3. Advanced Problem (30%)

LRU stands for Least-recently used, which is a replacement policy of choosing the one unused for the longest time. In this problem, you have to implement an n-way set-associative cache simulator **using LRU** (by C/C++, refer to the supplied file "direct_mapped_cache.cpp"). Please name this new simulator as "direct_mapped_cache_lru.cpp". Set block size = 64 bytes. Then input the file "LU.txt" and "RADIX.txt" that are the memory trace from two benchmarks to the simulator. Please draw a graph as the following example and describe the reason of rise and fall of the lines in the report. (Please separate the discussion of LU and RADIX). Also, please compute the total bits (including tags and one valid bit) required for each cache, and show them by table (like following table) or plot the figure.



Associativity / Cache size	1-way	2-way	4-way	8-way
1K				
2K				
4K				
8K				
16K				
32K				

4. Grade

a. Total score: 100%, **Copy will get 0 point!**

b. Basic score: 50%

c. Advanced score: 30%

d. Report: 20%

e. **Delay: 10% off/ day**

5. Hand in your Assignment

Please upload your assignment to E3.

Put all of *.v and *.cpp source files and report into the same compressed file. (Use your student ID to name your compressed file and it must be in the form of "student ID 1_student ID 2". For example, 0316001_0316002.zip)

Note: You must be uploading the ".zip" file to e3. Other filenames and formats such as *.rar and *.7z are NOT accepted!

6. Q&A

If you have any question, use E3 discussion or just send email to TAs.