CSE 378 Winter 1999

Machine Organization and Assembly Language Programming

# Programming Assignment and Problem Set #8

Due: Wednesday, December 8

## Programming Assignment

In this assignment you will write in SPIM the skeleton of a trace-driven simulator for assessing the performance of a cache.

Trace-driven simulation is a widely used technique to assess the performance of the components of the memory hierarchy. In broad terms, the simulator works as follows

**Input**

- A trace, i.e., a string, of memory references.
- One or more cache descriptions (I-cache, D-cache, cache hierarchy) with size, associativity, block size, replacement algorithm.
- A write policy.
- Access times of the various components of the memory hierarchy.

**Output**

- A set of statistics, e.g. hit ratio, or more precisely read hit ratio, write hit ratio, average memory access time etc.
- Cycles spent waiting at the various levels of the memory hierarchy etc.

**Algorithm**

- Process each memory reference in turn. Decompose the address in (tag, index, displacement) components
- Check if the memory reference hits in the appropriate cache (note that you don't bring data in the simulated cache; you only simulate the presence/absence of particular blocks)
- Take appropriate actions. In case of a hit, record statistics, maybe turn on some valid/dirty bits etc. In case of a miss, bring in the missing block, replace an old one, record statistics etc.

Your assignment is to write a cache simulator in SPIM with the following specifications.

**Input**

- The trace will consist of a string of data memory references (we will simulate only a D-cache) with the following format:$<$ *operation* $>$ $<$ *address* $>$ where *operation* is "1" for a read and "2" for a write, and *address* is a 32-bit byte address.

  The trace ends with the pair 0 0.

  An example trace will be given to you shortly. You can assume that it contains less than 1024 references.

- You'll have to simulate only a D-cache. Because the trace is small, we'll have a ridiculously small cache so that you don't have only compulsory misses.

  The D-cache is 128 bytes, 2-way set-associative, with a block size of 8 bytes, uses a write-back, write-allocate policy, and an LRU replacement algorithm.

**Output**
Your output, that should be displayed on the SPIM console, will consist of

- The number of data references that you processed.
- the number of hits in the D-cache
- the number of blocks you needed to write back.

For debugging purposes, you are strongly encouraged to test your program on smaller cache sizes and smaller input traces (e.g., using that of Problem 7.7 duly modified).

# Problem Set

1. Problems 7.33 and 7.34 (you'll have to read the text of Problem 7.32 but you don't have to give the answer to Problem 7.32)