CSE 378 Autumn 2005

Machine Organization & Assembly Language Programming

Problem Set #5

Due: 5 December 2005

In this assignment you will write in the C language a simple trace-driven simulator for assessing the performance of a cache. **This assignment is to be done individually, i.e. no partners.**

Trace-driven simulation is a widely used technique to assess the performance of the components of the memory hierarchy. For this assignment, the trace will consist of a string of data memory references (we will simulate only data caches) 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.

The basic idea is to process each memory reference in turn like we did in class. First you decompose the address in (tag,index,displacement) components. Then you check if there is a hit/miss (note that you don't need to simulate bringing data in the cache; it is sufficient to know which blocks are currently mapped in the cache). At this point, you need to record the statistics needed for the desired outputs and take appropriate actions (e.g., setting dirty bits, LRU information, replacements in case of a miss etc.).

Because the trace is short, we'll have a ridiculously small cache so that you don't have only compulsory misses. The caches that you'll have to consider are:

- Cache 1: 256 bytes, direct-mapped, line size 16 bytes. It uses a write-back, write-allocate policy.
- Cache 2: 256 bytes, 2-way set-associative, line size 16 bytes. It uses a write-back, write-allocate policy and an LRU replacement algorithm.
- Cache 3: Cache 1 backed up by a 3-entry victim cache using an LRU replacement algorithm.

The output statistics that you should generate are:

- The number of data references that you processed.
- For each of the 3 cache configurations, the number of hits in the cache (in the case of Cache 3, count separately the number of hits in the cache and the number of hits in the victim cache)
- For each of the 3 cache configurations, the number of blocks you needed to write back.
- Assuming that a hit in the cache takes 1 cycle, a hit in the victim cache takes 2 cycles, and a miss is resolved in 50 cycles, what are the average memory access times for the 3 configurations. (Don't draw any definite conclusion on the advantages/drawbacks of one configuration based on the results of this single minuscule and totally unrepresentative experiment!)

The trace can be found at:
http://www.cs.washington.edu/education/courses/378/04sp/homeworks/cactrace.txt

A C-skeleton for this program can be found at:
http://www.cs.washington.edu/education/courses/378/04sp/homeworks/cache.c

(A good warm-up would be to do Exercise 7.9 in your book by hand).