CS146: Computer Architecture
Fall 2019
Homework #4
Due Friday November 1, 2019

---

1. **Simple Caches**

   (a) Here is a string of address references given as word addresses: 1, 4, 8, 5, 20, 17, 19, 56, 9, 11, 4, 43, 5, 6, 9, 17. Assume a direct mapped cache with 16 one-word blocks that is initially empty, label each reference in the list as a hit or miss and show the final contents of the cache.

   (b) Using the same reference string, show the hits and misses and final cache contents for a direct mapped cache with four-word blocks and a total size of 16 words.

   (c) Using the same reference string, show the hits and misses and final cache contents for a two-way set associative cache with one-word blocks and a total size of 16 words. Assume LRU replacement.

   (d) Using the same reference string, show the hits and misses and final cache contents for a fully associative cache with one-word blocks and a total size of 16 words. Assume LRU replacement.

2. **H&P 5.3**

Make sure that you understand the basic concepts of the program in Exercise 5.2, although you do not have to implement and run the program for Exercise 5.3.

3. **Cache Design Experiments**

This problem gives you a feel for how quantitative evaluations and resource constraints can guide implementation choices in computer systems. In this problem, we will use SimpleScalar's "sim-cache" simulator that allows the user to model a cache memory system. Use the same benchmarks and script as for HW2 – choose 3 of the benchmarks. Hopefully you will only need to make very minor changes to the script to get things running with sim-cache. If you'd like, you can also use "sim-cheetah" which performs multiple cache simulations in parallel.

The quality of our cache organization will be determined by the average access time for that cache:

Average Access Time = Hit Time + Miss Rate * Miss Penalty

Where Miss Penalty = ((Block size in words)/2 + 10) * Hit Time

We will use a hit time of 1ns for a direct-mapped cache and 1.5ns for a set-associative cache.

You are given 80 4Kx8bit static RAMs for the cache, as well as a selection of miscellaneous logic parts. From this "inventory" you must be able to build up the entire cache, including the tags, data, and three control bits per cache line. The address supplied to the cache is 32-bits.

Your goal is to find the best cache organization for each of the benchmarks given. You should explore several different block sizes and several different degrees of associativity. (Reasonable degrees of associativity are from direct-mapped through 8-way set associative. Don't explore higher degrees than 8-way). Also, you can leave the TLB parameters unchanged – only look at L1 Dcache and L1 Icache.

Assume the caches are write-through and ignore all write stalls.

What to hand in:

You should create a summary of benchmark behavior on each cache configuration. The summary should include:

1) A cache description. This includes the size of the cache, the block size, the associativity. (The "cache size" is its true data capacity, NOT the amount of RAM that was used to build it including tags, etc).
2) A RAM accounting breakdown: This includes the amount of RAM used for the tags, data, and control bits of this organization.
3) The total miss rate, and the read and write miss rates for the benchmarks.
4) The average access time (as given by the formula above) per reference.

Also include an overall summary that gives your conclusions about the applications and cache behavior. Here, you should combine your observations about the different applications to make a recommendation about which cache structure would be best to build, given that the system obviously can't have a different cache for each application.