50.002 COMPUTATIONAL STRUCTURES

INFORMATION SYSTEMS TECHNOLOGY AND DESIGN

Problem Set 11

1 Problem 1

The following is a sequence of address references given as word addresses:

$$2, 3, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, 11$$

1. 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.

   Solution:
   2: miss, cache now holds: 2
   3: miss, cache now holds: 3, 2
   11: miss, cache now holds: 11, 3, 2
   16: miss, cache now holds: 16, 11, 3, 2
   21: miss, cache now holds: 21, 16, 11, 3, 2
   13: miss, cache now holds: 13, 21, 16, 11, 3, 2
   64: miss, cache now holds: 64, 13, 21, 16, 11, 3, 2
   48: miss, cache now holds: 48, 64, 13, 21, 16, 11, 3, 2
   19: miss, cache now holds: 19, 48, 64, 13, 21, 16, 11, 3, 2
   11: hit, cache now holds: 11, 19, 48, 64, 13, 21, 16, 3, 2
   3: hit, cache now holds: 3, 11, 19, 48, 64, 13, 21, 16, 2
   22: miss, cache now holds: 22, 3, 11, 19, 48, 64, 13, 21, 16, 2
   4: miss, cache now holds: 4, 22, 3, 11, 19, 48, 64, 13, 21, 16, 2
   27: miss, cache now holds: 27, 4, 22, 3, 11, 19, 48, 64, 13, 21, 16, 2
   6: miss, cache now holds: 6, 27, 4, 22, 3, 11, 19, 48, 64, 13, 21, 16, 2
   11: hit, cache now holds: 11, 6, 27, 4, 22, 3, 19, 48, 64, 13, 21, 16, 2

2. Show the hits and misses and final cache contents for a fully associative cache with four-word blocks and a total size of 16 words. Assume LRU replacement.
   Solution:
   With a N-word block of data for each cache entry, note that the N words in a

cache entry will have consecutive memory addresses *starting with a word address that's a multiple of N*.

2: miss, cache now holds: 0-3
3: hit, cache now holds: 0-3
11: miss, cache now holds: 8-11, 0-3
16: miss, cache now holds: 16-19, 8-11, 0-3
21: miss, cache now holds: 20-23, 16-19, 8-11, 0-3
13: miss, cache now holds: 12-15, 20-23, 16-19, 8-11
64: miss, cache now holds: 64-67, 12-15, 20-23, 16-19
48: miss, cache now holds: 48-51, 64-67, 12-15, 20-23
19: miss, cache now holds: 16-19, 48-51, 64-67, 12-15
11: miss, cache now holds: 8-11, 16-19, 48-51, 64-67
3: miss, cache now holds: 0-3, 8-11, 16-19, 48-51
22: miss, cache now holds: 20-23, 0-3, 8-11, 16-19
4: miss, cache now holds: 4-7, 20-23, 0-3, 8-11
27: miss, cache now holds: 24-27, 4-7, 20-23, 0-3
6: hit, cache now holds: 4-7, 24-27, 20-23, 0-3
11: miss, cache now holds: 8-11, 4-7, 24-27, 20-23

2 Problem 2

1. If a cache access requires one clock cycle and handling cache misses stalls the processor for an additional five cycles, which of the following cache hit rates comes closest to achieving an average memory access of 2 cycles?

   (a) 75%
   (b) 80%
   (c) 83%
   (d) 86%
   (e) 98%

   Solution:
   2 cycle average access = (1 cycle for cache) + (1 - hit rate)(5 cycles stall). This means hit rate = 80% (B)

2. LRU is an effective cache replacement strategy primarily because programs

   (a) Exhibit locality of reference
   (b) Usually have small working sets
   (c) Read data much more frequently than write data

   Solution:

Locality implies that the probability of accessing a location decreases as the time since the last access increases. By choosing to replace locations that haven't been used for the longest time, the least-recently-used replacement strategy should, in theory, be replacing locations that have the lowest probability of being accessed in the future.

3. If increasing the block size of a cache improves performance it is primarily because programs

   (a) Exhibit locality of reference

   (b) Usually have small working sets

   (c) Read data much more frequently than write data

Solution:
(A). Increased block size means that more words are fetched when filling a cache line after a miss on a particular location. If this leads to increased performance, then the nearby words in the block must have been accessed by the program later on, ie, the program is exhibiting locality.

3 Problem 3

A student has miswired the address lines going to the memory of an unpipelined BETA. The wires in question carry a 30-bit word address to the memory subsystem, and the hapless student has in fact reversed the order of all 30 address bits. Much to his surprise, the machine continues to work perfectly.

1. Explain why the miswiring doesn't affect the operation of the machine.

   Solution:
   Since the Beta reverses the order of the 30 bit address in the same manner for each memory access, the Beta will use the same reversed address to access a particular memory location for both stores and loads. Thus, the operation of the machine will not be affected.

2. The student now replaces the memory in his miswired BETA with a supposedly higher performance unit that contains both a fast fully associative cache and the same memory as before. The reversed wiring still exists between the BETA and this new unit. To his surprise, the new unit does not significantly improve the performance of his machine. In desperation, the student then fixes the reversal of his address lines and the machine's performance improves tremendously. Explain why this happens.

   Solution:
   Caches take advantage of locality of reference by reading in an entire block of

related data at one time, thereby reducing main memory accesses. By reversing the order of the 30 bit address, locality of the memory addresses is disrupted. The low-order bits that would normally place related data close to one another are instead the high-order bits and related data is more spread out through the main memory. This reduction in locality reduces cache performance significantly. When the student fixes the address line reversal problem, locality of the memory is restored, and the cache can perform as intended.

4 Problem 4

The following questions ask you to evaluate alternative cache designs using patterns of memory references taken from running programs. Each of the caches under consideration has a total capacity of 8 (4-byte) words, with one word stored in each cache line. The cache designs under consideration are:

- DM: a direct-mapped cache.
- S2: a 2-way set-associative cache with a least-recently-used replacement policy.
- FA: a fully-associative cache with a least-recently-used replacement policy.

The questions below present a sequence of addresses for memory reads. **You should assume the sequences repeat from the start whenever you see "..."**. Keep in mind that byte addressing is used; addresses of consecutive words in memory differ by 4. Each question asks which cache(s) give the best hit rate for the sequence. Answer by considering the steady-state hit rate, i.e., the percentage of memory references that hit in the cache after the sequence has been repeated many times.

1. Which cache(s) have the best hit rate for the sequence 0, 16, 4, 36, ...

   Solution:
   DM: locations 4 and 36 collide, so each iteration has 2 hits, 2 misses.
   S2: 100% hit rate. 0 and 16 map to the same cache line, as do 4 and 36, but since the cache is 2-way associative they don't collide.
   FA: 100% hit rate. The cache is only half filled by this loop.

2. Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 16, 20, 24, 28, 32, ...

   Solution:
   DM: locations 0 and 32 collide, so each iteration has 7 hits, 2 misses.
   S2: locations 0, 16 and 32 all map to the same cache line. The LRU replacement strategy replaces 0 when accessing 32, 16 when accesing 0, 32 when accessing 16, etc., so each iteration has 6 hits, 3 misses.
   FA: has 0% hit rate in the steady state since the LRU replacement strategy throws out each location just before it's accessed by the loop

3. Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 16, 20, 24, 28, 32, 28, 24, 20, 16, 12, 8, 4, ...

   Solution:
   All caches perform the same – locations 0 and 32 trade places in the caches, so each iteration has 14 hits and 2 misses.

4. Which cache(s) have the best hit rate for the sequence 0, 4, 8, 12, 32, 36, 40, 44, 16, ..

   Solution:
   DM: 32 collides with 0, 36 with 4, 40 with 8, 44 with 12, so each itreation has only 1 hit and 8 misses.
   S2: locations 0, 16 and 32 trade places in the cache, so each iteration has 6 hits and 3 misses.
   FA: 0 hits since LRU throws out each location just before it's accessed by the loop.

5. Assume that a cache access takes 1 cycle and a memory access takes 4 cycles. If a memory access is initiated only after the cache has missed, what is the maximum miss rate we can tolerate before use of the cache actually slows down accesses?

   Solution:
   If accesses always go to memory, it takes 4 cycles per access. Using the cache the average number of cycles per access is 1 + (miss rate)*4.
   So if the miss rate is larger than 75% the average number of cycles per access is more than 4.