Memory Hierarchy Assignment (Learning)

CSCI 389: Computer Systems

Spring 2022

This assignment is an opportunity to test your understanding of the memory hierarchy and receive feedback. Point values are assigned so that you can differentiate between large and small mistakes, but this assignment does not affect your grade.

Due Date: Friday, February 25th at 10:00 am.

1. (10 points) **Average Memory Access Time.** Consider a system with an L1 cache and an L2 cache between the processor and memory. The system has the following characteristics:

   - The L1 cache takes 1 cycle to access.
   - The L1 cache has a 5% miss ratio.
   - The L2 cache takes 15 cycle to access.
   - The L2 cache has a 30% miss ratio.
   - Memory takes 180 cycles to access.

   (a) (6 points) What is the average memory access time for this system?

   (b) (4 points) What is the average memory access time if the miss ratio of the L1 cache increases to 10%?

   A.

   - Memory takes 180 cycles
   - L2 Cache takes $15 + 0.3 \cdot 180 = 15 + 54 = 69$
   - L1 cache takes $1 + 0.05 \cdot 69 = 1 + 3.45 = 4.45$ this is rounded up to 5

   B.

   - Memory takes 180 cycles
   - L2 Cache takes $15 + 0.3 \cdot 180 = 15 + 54 = 69$
   - L1 cache takes $1 + 0.1 \cdot 69 = 1 + 6.9 = 7.9$ this is rounded up to 8

2. (20 points) **Caching Bits.** Consider the following request sequence.

$$0x1000, 0x1100, 0x1008, 0x1108, 0x1010, 0x1110, 0x2000, 0x2100$$

   (a) (5 points) Provide a physical diagram showing how a direct-mapped cache with block size of 16 bytes will cache this sequence. Assume that the cache can store 128 bytes of data and runs LRU.

   (b) (5 points) Provide a physical diagram showing how a 2-way set associative cache with block size of 16 bytes will cache this sequence. Assume that the cache can store 128 bytes of data and runs LRU.

   (c) (5 points) Provide a physical diagram showing how a fully associative cache with block size of 16 bytes will cache this sequence. Assume that the cache can store 128 bytes of data and runs LRU.

   (d) (5 points) Provide a physical diagram showing how a fully associative cache with block size of 32 bytes will cache this sequence. Assume that the cache can store 128 bytes of data and runs LRU.

A.

Request 1: $0x1000 = 0001000000000000$

valid	tag	Set Index
1	1 0000 0000 0000	
0	001	
0	010	
0	011	
0	100	
0	101	
0	110	
0	111	

3. (20 points) **Eviction Policies.** Consider the following request sequence:

$$A, B, C, A, D, E, A, B, C$$

(a) (5 points) Provide a logical diagram that shows how an optimal fully associative cache of size 3 will cache this sequence.

(b) (5 points) Provide a logical diagram that shows how a FIFO fully associative cache of size 3 will cache this sequence.

(c) (5 points) Provide a logical diagram that shows how an LRU fully associative cache of size 3 will cache this sequence.

(d) (5 points) Label each miss in your diagram for part c with its type.

A.
Assuming there are no pre-fetches, then this is the ideal structure:

Cache	Req 1 (A)	Req 2 (B)	Req 3 (C)	Req 4 (A)	Req 5 (D)	Req 6 (E)	Req 7 (A)	Req 8 (B)	Req 9 (C)
1	A	A	A	A	A	A	A	A	A
2	*Empty*	B	B	B	B	B	B	B	B
3	*Empty*	*Empty*	C	C	D	E	E	E	C

This is due to the fact that D & E are only requested once while A, B, & C are all requested at least twice.

Because bypassing is not mentioned (and therefore presumably banned) The third cache storage is used

B.

Cache	Req 1 (A)	Req 2 (B)	Req 3 (C)	Req 4 (A)	Req 5 (D)	Req 6 (E)	Req 7 (A)	Req 8 (B)	Req 9 (C)
1	A (1)	A (1)	A (1)	A (1)	D (5)	D (5)	D (5)	B (8)	B (8)
2	*Empty*	B (2)	B (2)	B (2)	B (2)	E (6)	E (6)	E (6)	C (9)
3	*Empty*	*Empty*	C (3)	C (3)	C (3)	C (3)	A (7)	A (7)	A (7)

C.

Cache	Req 1 (A)	Req 2 (B)	Req 3 (C)	Req 4 (A)	Req 5 (D)	Req 6 (E)	Req 7 (A)	Req 8 (B)	Req 9 (C)
1	A (1)	A (1)	A (1)	A (4)	A (4)	A (4)	A (7)	A (7)	A (7)
2	*Empty*	B (2)	B (2)	B (2)	D (5)	D (5)	D (5)	B (8)	B (8)
3	*Empty*	*Empty*	C (3)	C (3)	C (3)	E (6)	E (6)	E (6)	C(9)

D.

- **Steps 1-3**: Compulsory misses (First time seeing A, B, & C)

- **Step 5-6**: Compulsory misses (although the cache blocks that they overwrite are used later the addresses have not been previously seen)
- **Step 8-9**: Replacement misses (if D & E had been bypassed then B & C would not need to be fetched again)