Memory Hierarchy Assignment (Evaluation)

CSCI 389: Computer Systems

Spring 2022

This assignment is designed to test your understanding of the memory hierarchy. Feel free to collaborate wih others and use resources, but all code and writeups must be your own.

Due Date: Friday, March 4th 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 2 cycles to access.
   - The L1 cache has a 10% miss ratio.
   - The L2 cache takes 20 cycle to access.
   - The L2 cache has a 15% miss ratio.
   - Memory takes 220 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 decreases to 5%?

   A.

   - Memory takes 220 cycles
   - L2 takes $20 + 0.15(220) = 53$
   - L1 takes $2 + 0.1(53) = 7.3$

   Thus the average memory access time is 7.3 cycles

   B.

   - Memory takes 220 cycles
   - L2 takes $20 + 0.15(220) = 53$ cycles on average.
   - L1 takes $2 + 0.05(53) = 4.65$ cycles on average.

   Thus the average memory access time is 4.65 cycles

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

   $$0x0000, 0x8400, 0x6082, 0x840C, 0x8411, 0x2148, 0x2110, 0x2420, 0x2100, 0x8403$$

   For each part, you will draw a physical diagram, which consists of the tags and valid bits for each cache line. Clearly mark which cache lines are in which sets. In the area where data would normally go, simply denote the range of addresses covered.

   (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.

Hex	**Tag(First 9)**	**Set (Middle 3)**	**Offset (Last 4 bits)**
0x0000	0000 0000 0	000	0000
0x8400	1000 0100 0	000	0000
0x6082	0110 0000 1	000	0010
0x840C	1000 0100 0	000	1100
0x8411	1000 0100 0	001	0001
0x2148	0010 0001 0	100	1000
0x2420	0010 0100 0	010	0000
0x2100	0010 0001 0	000	0000
0x8403	1000 0100 0	000	0011

Tag	**Set**	**Offset**
000	∅	∅
001	∅	∅
010	∅	∅
011	∅	∅
100	∅	∅
101	∅	∅
110	∅	∅
111	∅	∅

$\longrightarrow$

Tag	**Set**	**Offset**
000	0000 0000	0000
001	∅	∅
010	∅	∅
011	∅	∅
100	∅	∅
101	∅	∅
110	∅	∅
111	∅	∅

$\longrightarrow$

Tag	**Set**	**Offset**
000	1000 0100 0	0000
001	∅	∅
010	∅	∅
011	∅	∅
100	∅	∅
101	∅	∅
110	∅	∅
111	∅	∅

$\longrightarrow$

Tag	**Set**	**Offset**
000	0110 0000 1	0010
001	∅	∅
010	∅	∅
011	∅	∅
100	∅	∅
101	∅	∅
110	∅	∅
111	∅	∅

$\longrightarrow$

Tag	**Set**	**Offset**
000	0110 0000 1	0010
001	0110 0000 1	0001
010	∅	∅
011	∅	∅
100	∅	∅
101	∅	∅
110	∅	∅
111	∅	∅

$\longrightarrow$

Tag	**Set**	**Offset**
000	0110 0000 1	0010
001	0110 0000 1	0001
010	∅	∅
011	∅	∅
100	0010 0001 0	1000
101	∅	∅
110	∅	∅
111	∅	∅

Tag	**Set**	**Offset**
000	0110 0000 1	0010
001	0110 0000 1	0001
010	0010 0100 0	0000
011	∅	∅
100	0010 0001 0	1000
101	∅	∅
110	∅	∅
111	∅	∅

$\longrightarrow$

Tag	**Set**	**Offset**
000	0010 0001 0	0000
001	0110 0000 1	0001
010	0010 0100 0	0000
011	∅	∅
100	0010 0001 0	1000
101	∅	∅
110	∅	∅
111	∅	∅

$\longrightarrow$

Tag	**Set**	**Offset**
000	1000 0100 0	0011
001	0110 0000 1	0001
010	0010 0100 0	0000
011	∅	∅
100	0010 0001 0	1000
101	∅	∅
110	∅	∅
111	∅	∅

B.

Hex	Tag(First 10)	Set (Middle 2)	Offset (Last 4 bits)
0x0000	0000 0000 00	00	0000
0x8400	1000 0100 00	00	0000
0x6082	0110 0000 10	00	0010
0x840C	1000 0100 00	00	1100
0x8411	1000 0100 00	01	0001
0x2148	0010 0001 01	00	1000
0x2420	0010 0100 00	10	0000
0x2100	0010 0001 00	00	0000
0x8403	1000 0100 00	00	0011

Tag	Set	Offset
00	0000 0000 00	0000
01	∅	∅
10	∅	∅
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	1000 0100 00	0000
01	∅	∅
10	∅	∅
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	0110 0000 10	0010
01	∅	∅
10	∅	∅
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	1000 0100 00	1100
01	∅	∅
10	∅	∅
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	1000 0100 00	1100
01	1000 0100 00	0001
10	∅	∅
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	0010 0001 01	1000
01	1000 0100 00	0001
10	∅	∅
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	0010 0001 01	1000
01	1000 0100 00	0001
10	0010 0100 00	0000
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	0010 0001 01	0000
01	1000 0100 00	0001
10	0010 0100 00	0000
11	∅	∅

$\longrightarrow$

Tag	Set	Offset
00	1000 0100 00	0011
01	1000 0100 00	0001
10	0010 0100 00	0000
11	∅	∅

C.

Hex	Tag(First 12)	Offset (Last 4 bits)
0x0000	0000 0000 0000	0000
0x8400	1000 0100 0000	0000
0x6082	0110 0000 1000	0010
0x840C	1000 0100 0000	1100
0x8411	1000 0100 0001	0001
0x2148	0010 0001 0100	1000
0x2420	0010 0100 0010	0000
0x2100	0010 0001 0000	0000
0x8403	1000 0100 0000	0011

Tag(First 12)	Offset (Last 4 bits)
0000 0000 0000	0000
∅	∅
∅	∅
∅	∅
∅	∅
∅	∅
∅	∅
∅	∅

$\rightarrow$

Tag(First 12)	Offset (Last 4 bits)
0000 0000 0000	0000
1000 0100 0000	0000
∅	∅
∅	∅
∅	∅
∅	∅
∅	∅
∅	∅

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
1000 0100 0000	0000
0110 0000 1000	0010
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
0110 0000 1000	0010
1000 0100 0000	1100
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
0110 0000 1000	0010
1000 0100 0000	1100
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
0110 0000 1000	0010
1000 0100 0000	1100
1000 0100 0001	0001
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
0110 0000 1000	0010
1000 0100 0000	1100
1000 0100 0001	0001
0010 0001 0100	1000
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
0110 0000 1000	0010
1000 0100 0000	1100
1000 0100 0001	0001
0010 0001 0100	1000
0010 0100 0010	0000
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
0110 0000 1000	0010
1000 0100 0000	1100
1000 0100 0001	0001
0010 0001 0100	1000
0010 0100 0010	0000
0010 0001 0000	0000
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 0000	0000
0110 0000 1000	0010
1000 0100 0000	1100
1000 0100 0001	0001
0010 0001 0100	1000
0010 0100 0010	0000
0010 0001 0000	0000
1000 0100 0000	0011

D.

Hex	**Tag(First 11)**	**Offset (Last 5 bits)**
0x0000	0000 0000 000	0 0000
0x8400	1000 0100 000	0 0000
0x6082	0110 0000 100	0 0010
0x840C	1000 0100 000	0 1100
0x8411	1000 0100 000	1 0001
0x2148	0010 0001 010	0 1000
0x2420	0010 0100 001	0 0000
0x2100	0010 0001 000	0 0000
0x8403	1000 0100 000	0 0011

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
0110 0000 100	0 0010
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
0110 0000 100	0 0010
1000 0100 000	0 1100
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
0110 0000 100	0 0010
1000 0100 000	1 0001
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
0110 0000 100	0 0010
1000 0100 000	1 0001
0010 0001 010	0 1000
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
0110 0000 100	0 0010
1000 0100 000	1 0001
0010 0001 010	0 1000
0010 0100 001	0 0000
$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
0110 0000 100	0 0010
1000 0100 000	1 0001
0010 0001 010	0 1000
0010 0100 001	0 0000
0010 0001 000	0 0000
$\emptyset$	$\emptyset$

$\rightarrow$

Tag(First 12)	**Offset (Last 4 bits)**
0000 0000 000	0 0000
1000 0100 000	0 0000
0110 0000 100	0 0010
1000 0100 000	1 0001
0010 0001 010	0 1000
0010 0100 001	0 0000
0010 0001 000	0 0000
1000 0100 000	0 0011

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

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

   (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.

For all sub questions **bold represents replacement** and *italics represents hit*

A.

Cache	1 (A)	2 (B)	3 (C)	4 (B)	5 (A)	6 (D)	7 (C)	8 (E)	9 (B)	10 (D)	11 (A)	12 (C)
1	**A**	A	A	A	*A*	**D**	D	D	D	*D*	**A**	**C**
2	∅	**B**	B	*B*	B	B	B	B	*B*	B	B	B
3	∅	∅	**C**	C	C	C	*C*	**E**	E	E	E	E

B.

Cache	1 (A)	2 (B)	3 (C)	4 (B)	5 (A)	6 (D)	7 (C)	8 (E)	9 (B)	10 (D)	11 (A)	12 (C)
1	**A (1)**	A (1)	A (1)	A (1)	*A (1)*	**D (6)**	D (6)	D (6)	D (6)	*D (6)*	**A (11)**	A (11)
2	∅	**B (2)**	B (2)	*B (2)*	B (2)	B (2)	B (2)	**E (8)**	E (8)	E (8)	E (8)	**C (12)**
3	∅	∅	**C (3)**	C (3)	C (3)	C (3)	*C (3)*	C (3)	**B (9)**	B (9)	B (9)	B (9)

C.

Cache	1 (A)	2 (B)	3 (C)	4 (B)	5 (A)	6 (D)	7 (C)	8 (E)	9 (B)	10 (D)	11 (A)	12 (C)
1	**A (1)**	A (1)	A (1)	A (1)	*A (5)*	A (5)	A (5)	**E (8)**	E (8)	E (8)	**A (11)**	A (11)
2	∅	**B (2)**	B (2)	*B (4)*	B (4)	B (4)	**C (7)**	C (7)	C (7)	**D (10)**	D (10)	D (10)
3	∅	∅	**C (3)**	C (3)	C (3)	**D (6)**	D (6)	D (6)	**B (9)**	B (9)	B (9)	**C (12)**

D.

- Requests 1-3: Compulsory misses
- Request 6: Compulsory miss
- Request 7: Replacement miss
- Request 8: Compulsory miss
- Request 9-10: Replacement miss
- Request 11-12: Capacity miss