Computer Architecture

Ali Muhammad
aa07190

Homework 4

Question:	1	2	3	4	5	6	7	8	Total
Points:	10	10	15	5	10	10	20	20	100
Score:									

Question 1 [10 Marks]

Caches are important to providing a high-performance memory hierarchy to processors. Below is a list of 64-bit memory address references, given as word addresses.
0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5, 0x2c, 0xba, 0xfd

(a) **[05 Marks]** For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty. First insertion is already done so that you may get the idea.

Word Address	**Binary Address**	**Tag**	**Index**	**Hit/Miss**
0x03	0000 0011	0	3	M
0xb4	1011 0100	b	4	M
0x2b	0010 1011	2	b	M
0x02	0000 0010	0	2	M
0xbf	1011 1111	b	f	M
0x58	0101 1000	5	8	M
0xbe	1011 1110	b	e	M
0x0e	0000 1110	0	e	M
0xb5	1011 0101	b	5	M
0x2c	0010 1100	2	c	M
0xba	1011 1010	b	a	M
0xfd	1111 1101	f	d	M

(b) **[05 Marks]** For each of these references, identify the binary word address, the tag, the index, and the offset given a direct-mapped cache with two-word blocks and a total size of eight blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Word Address	Binary Address	Tag	Index	Offset	Hit/Miss
0x03	0000 0011	0	1	1	M
0xb4	1011 0100	b	2	0	M
0x2b	0010 1011	2	5	1	M
0x02	0000 0010	0	1	0	H
0xbf	1011 1111	b	7	1	M
0x58	0101 1000	5	4	0	M
0xbe	1011 1110	b	7	0	H
0x0e	0000 1110	0	7	0	M
0xb5	1011 0101	b	2	1	H
0x2c	0010 1100	2	6	0	M
0xba	1011 1010	b	5	0	M
0xfd	1111 1101	f	6	1	M

Question 2 [10 Marks]

For a direct-mapped cache design with a 64-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
63-10	9-5	4-0

(a) **[03 Marks]** What is the cache block size (in words)?

Solution: Offset = 5bits. Block Size = $2^5 = 32$

Block Size (Words) = $\frac{32\text{bytes}}{8\text{bytes}} = 4$ words.

(b) **[03 Marks]** How many blocks does the cache have?

Solution: Index Bits = 5 $\implies$ Blocks = $2^5 = 32$ blocks.

(c) **[04 Marks]** What is the ratio between the total bits required for such a cache implementation over the data storage bits?

Solution: Ratio = $\frac{\text{Tag Bits} + \text{Valid Bits} + \text{Data Bits}}{\text{Data Bits}}$

Ratio = $\frac{(54 \times 32) + (1 \times 32) + (32 \times 4 \times 8 \times 8)}{(32 \times 4 \times 8 \times 8)} = 1.21$

Question 3 [15 Marks]

Considering the address size of 64-bits, fill in the data for difference types of caches:

Cache Type	Blocks	Data per block	Sets	Associativity - ways	Tag Bits	Index Bits	Offset Bits
Fully Associative	8	8 words	- -	8	59	- -	5
Direct Mapped	16	8 words	- -	1	55	4	5
Set Associative	32	8 words	4	8	57	2	5
Direct Mapped	64	8 words	- -	1	53	6	5
Set Associative	128	8 words	32	4	54	5	5
Set Associative	256	8 words	32	8	54	5	5
Fully Associative	512	8 words	- -	512	59	- -	5
Direct Mapped	1024	8 words	- -	1	49	10	5
Set Associative	2048	8 words	64	32	53	6	5
Direct Mapped	4096	8 words	- -	1	47	12	5

Question 4 [05 Marks]

Assume the miss rate of an instruction cache is 4% and the miss rate of the data cache is 6%. If a processor has a CPI of 3 without any memory stalls, and the miss penalty is 100 cycles for all misses, determine how much faster a processor would run with a perfect cache that never missed. Assume the frequency of all loads and stores is 26%.

Solution:

Given:
$\text{miss-rate}_{\text{I-cache}} = 4\%$
$\text{miss-rate}_{\text{D-cache}} = 6\%$
$\text{CPI}_{\text{perfect}} = 3$
miss penalty = 100 cycles
load/store instructions = 26%

CPI_{miss} :
I-cache = miss $\text{rate}_{\text{I-cache}} *$ miss penalty $*$ instruction memory access / program
$= 0.04 * 100 * 1 = 4$
D-cache = miss $\text{rate}_{\text{D-cache}} *$ miss penalty $*$ data memory access / program
$= 0.06 * 100 * 0.26 = 1.56$

$\text{CPI}_{\text{stall}} = \text{CPI}_{\text{perfect}} + \text{CPI}_{\text{miss}} = 3 + 4 + 1.56 = 8.56$

$$\frac{\text{CPU time with stalls}}{\text{CPU time with perfect cache}} = \frac{\text{I} * \text{CPI}_{\text{stall}} * \text{Clock cycle}}{\text{I} * \text{CPI}_{\text{perfect}} * \text{Clock cycle}} = \frac{\text{CPI}_{\text{stall}}}{\text{CPI}_{\text{perfect}}} = \frac{8.56}{3} = 2.853$$

Question 5 [10 Marks]

We are given 4 arrays of size 6. Each element in an array is of 32 bytes i.e., one word. Following is the data stored in the array:

A = (25, 48, 43, 30, 47, 36)
B = (16, 29, 35, 38, 32, 41)
C = (24, 33, 5, 39, 10, 14)
D = (23, 7, 11, 44, 42, 22)

The array data is arranged in main memory as follows:

00000	A[0]
00001	A[1]
00010	A[2]
00011	A[3]
00100	A[4]
00101	A[5]
00110	
00111	
01000	B[0]
01001	B[1]
01010	B[2]
01011	B[3]
01100	B[4]
01101	B[5]
01110	
01111	
10000	C[0]
10001	C[1]
10010	C[2]
10011	C[3]
10100	C[4]
10101	C[5]
10110	
10111	
11000	D[0]
11001	D[1]
11010	D[2]
11011	D[3]
11100	D[4]
11101	D[5]
11110	
11111	

We are given a direct mapped cache which contains 8 block (each block will contain one word). Insert the following elements in cache one by one and also mention whether it ws a hit or a

miss. Assume that the first block of the cache will be populated by the first element of the array and so on. First insertion is already done so that you may get the idea.

Data to be Inserted	**Hit/Miss**	**Cache Index**							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A[0]	M	A[0]							
A[1]	M	A[0]	A[1]						
A[2]	M	A[0]	A[1]	A[2]					
A[1]	H	A[0]	A[1]	A[2]					
A[5]	M	A[0]	A[1]	A[2]			A[5]		
B[5]	M	A[0]	A[1]	A[2]			B[5]		
B[4]	M	A[0]	A[1]	A[2]		B[4]	B[5]		
B[3]	M	A[0]	A[1]	A[2]	B[3]	B[4]	B[5]		
B[3]	H	A[0]	A[1]	A[2]	B[3]	B[4]	B[5]		
B[4]	H	A[0]	A[1]	A[2]	B[3]	B[4]	B[5]		
D[1]	M	A[0]	D[1]	A[2]	B[3]	B[4]	B[5]		
D[2]	M	A[0]	D[1]	D[2]	B[3]	B[4]	B[5]		
D[3]	M	A[0]	D[1]	D[2]	D[3]	B[4]	B[5]		
D[4]	M	A[0]	D[1]	D[2]	D[3]	D[4]	B[5]		
C[3]	M	A[0]	D[1]	D[2]	C[3]	D[4]	B[5]		
C[2]	M	A[0]	D[1]	C[2]	C[3]	D[4]	B[5]		
C[4]	M	A[0]	D[1]	C[2]	C[3]	C[4]	B[5]		
C[2]	H	A[0]	D[1]	C[2]	C[3]	C[4]	B[5]		

What is the Hit Ratio and the Miss Ratio in the above case?

Solution: Total Hits = 4, Total Misses = 14, Total Accesses = 18.

Hit Ratio $= \frac{4}{18} = \frac{2}{9} = 0.222$; Miss Ratio $= \frac{14}{18} = \frac{7}{9} = 0.778$

Question 6 [10 Marks]

Whenever an element from an array is accessed, it is most probable that some other remaining elements of the array are also accessed. Repeat the same task as in Question 5 but this time design a cache with 4 blocks in which each block can accommodate 2 words. The first insertion is done again so that you may get the idea.

Data to be Inserted	**Hit/Miss**	**Cache Index**							
		0		**1**		**2**		**3**	
A[0]	M	A[0]	A[1]						
A[1]	H	A[0]	A[1]						
A[2]	M	A[0]	A[1]	A[2]	A[3]				
A[1]	H	A[0]	A[1]	A[2]	A[3]				
A[5]	M	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]		
B[5]	M	A[0]	A[1]	A[2]	A[3]	B[4]	B[5]		
B[4]	H	A[0]	A[1]	A[2]	A[3]	B[4]	B[5]		
B[3]	M	A[0]	A[1]	B[2]	B[3]	B[4]	B[5]		
B[3]	H	A[0]	A[1]	B[2]	B[3]	B[4]	B[5]		
B[4]	H	A[0]	A[1]	B[2]	B[3]	B[4]	B[5]		
D[1]	M	D[0]	D[1]	B[2]	B[3]	B[4]	B[5]		
D[2]	M	D[0]	D[1]	D[2]	D[3]	B[4]	B[5]		
D[3]	H	D[0]	D[1]	D[2]	D[3]	B[4]	B[5]		
D[4]	M	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]		
C[3]	M	D[0]	D[1]	C[2]	C[3]	D[4]	D[5]		
C[2]	H	D[0]	D[1]	C[2]	C[3]	D[4]	D[5]		
C[4]	M	D[0]	D[1]	C[2]	C[3]	C[4]	C[5]		
C[2]	H	D[0]	D[1]	C[2]	C[3]	C[4]	C[5]		

What is the Hit Ratio and the Miss Ratio in this case? Is it better than the previous? Does loading the whole array into the cache helps us in accessing the elements fast?

Solution: Total Hits = 8, Total Misses = 10, Total Accesses = 18.
Hit Ratio = $\frac{8}{18} = \frac{4}{9} = 0.444$, Miss Ratio = $\frac{10}{18} = \frac{5}{9} = 0.556$.
This is better than the previous one as we have more hits and less misses.

Loading the whole array would no doubt help us in accessing the elements fast, as long as the cache has enough capacity to accomodate the entire array, and the array is accessed more in the near future. Since cache memory is faster than main memory, it would reduce the overall latency for memory access in case the elements of the array are not in the cache.

Question 7 [20 Marks]

(a) **[05 Marks]** Repeat Question 5, this time using a fully associative cache containing 8 blocks. Use LRU replacement scheme for eviction.

Data to be Inserted	**Hit/Miss**	**Cache Index**							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A[0]	M	A[0]							
A[1]	M	A[0]	A[1]						
A[2]	M	A[0]	A[1]	A[2]					
A[1]	H	A[0]	A[2]	A[1]					
A[5]	M	A[0]	A[2]	A[1]	A[5]				
B[5]	M	A[0]	A[2]	A[1]	A[5]	B[5]			
B[4]	M	A[0]	A[2]	A[1]	A[5]	B[5]	B[4]		
B[3]	M	A[0]	A[2]	A[1]	A[5]	B[5]	B[4]	B[3]	
B[3]	H	A[0]	A[2]	A[1]	A[5]	B[5]	B[4]	B[3]	
B[4]	H	A[0]	A[2]	A[1]	A[5]	B[5]	B[3]	B[4]	
D[1]	M	A[0]	A[2]	A[1]	A[5]	B[5]	B[3]	B[4]	D[1]
D[2]	M	A[2]	A[1]	A[5]	B[5]	B[3]	B[4]	D[1]	D[2]
D[3]	M	A[1]	A[5]	B[5]	B[3]	B[4]	D[1]	D[2]	D[3]
D[4]	M	A[5]	B[5]	B[3]	B[4]	D[1]	D[2]	D[3]	D[4]
C[3]	M	B[5]	B[3]	B[4]	D[1]	D[2]	D[3]	D[4]	C[3]
C[2]	M	B[3]	B[4]	D[1]	D[2]	D[3]	D[4]	C[3]	C[2]
C[4]	M	B[4]	D[1]	D[2]	D[3]	D[4]	C[3]	C[2]	C[4]
C[2]	H	B[4]	D[1]	D[2]	D[3]	D[4]	C[3]	C[4]	C[2]

(b) **[05 Marks]** Repeat Question 6, this time using a fully associative cache containing 4 blocks in which each block can accomodate 2 words. Use LRU replacement scheme for eviction.

Data to be Inserted	**Hit/Miss**	**Cache Index**							
		0		**1**		**2**		**3**	
A[0]	M	A[0]	A[1]						
A[1]	H	A[0]	A[1]						
A[2]	M	A[0]	A[1]	A[2]	A[3]				
A[1]	H	A[2]	A[3]	A[0]	A[1]				
A[5]	M	A[2]	A[3]	A[0]	A[1]	A[4]	A[5]		
B[5]	M	A[2]	A[3]	A[0]	A[1]	A[4]	A[5]	B[4]	B[5]
B[4]	H	A[2]	A[3]	A[0]	A[1]	A[4]	A[5]	B[4]	B[5]
B[3]	M	A[0]	A[1]	A[4]	A[5]	B[4]	B[5]	B[2]	B[3]
B[3]	H	A[0]	A[1]	A[4]	A[5]	B[4]	B[5]	B[2]	B[3]
B[4]	H	A[0]	A[1]	A[4]	A[5]	B[2]	B[3]	B[4]	B[5]
D[1]	M	A[4]	A[5]	B[2]	B[3]	B[4]	B[5]	D[0]	D[1]
D[2]	M	B[2]	B[3]	B[4]	B[5]	D[0]	D[1]	D[2]	D[3]
D[3]	H	B[2]	B[3]	B[4]	B[5]	D[0]	D[1]	D[2]	D[3]
D[4]	M	B[4]	B[5]	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]
C[3]	M	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	C[2]	C[3]
C[2]	H	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	C[2]	C[3]
C[4]	M	D[2]	D[3]	D[4]	D[5]	C[2]	C[3]	C[4]	C[5]
C[2]	H	D[2]	D[3]	D[4]	D[5]	C[2]	C[3]	C[4]	C[5]

(c) **[05 Marks]** Compare the Hit & Miss Ratio for both the cases (i.e., loading a single element vs. loading two elements) for both the caches. Which cache helps us in accessing elements faster? and in which case? [Hint: To answer which cache helps us in accessing the elements faster, compare the Hit Ratios for both the caches]

Solution:

(a) Hits = 4, Misses = 14, Total Accesses = 18.
Hit Ratio = $\frac{4}{18} = \frac{2}{9}$, Miss Ratio = $\frac{14}{18} = \frac{7}{9}$. The hit and miss ratio for a Fully Associative Cache with LRU Replacement is same as for a Direct Mapped Cache.

(b) Hits = 8, Misses = 10, Total Accesses = 18.
Hit Ratio = $\frac{8}{18} = \frac{4}{9}$, Miss Ratio = $\frac{10}{18} = \frac{5}{9}$. The hit and miss ratio for a Fully Associative Cache containing 4 blocks each accomodating 2 words with LRU Replacement is the same as for a Direct Mapped Cache containing 4 blocks each accomodating 2 words.

However, in the case when we load two elements together, we get more hits as compared to loading a single element. For the given question, it doesn't matter which cache we use, however, loading two elements would result in faster access of elements as we get more hits, so with either cache, case 2 (loading two elements) should be used.

(c) **[05 Marks]** Repeat Question 6, this time using a two-way set associative cache with 2 sets of 2 blocks. Use LRU replacement scheme for eviction.

Data to be Inserted	**Hit/Miss**	**Cache Index**			
		0		**1**	
A[0]	M	A[0]	A[1]		
A[1]	H	A[0]	A[1]		
A[2]	M	A[0]	A[1]	A[2]	A[3]
A[1]	H	A[0]	A[1]	A[2]	A[3]
A[5]	M	A[0]	A[1]	A[4]	A[5]
B[5]					
B[4]					
B[3]					
B[3]					
B[4]					
D[1]					
D[2]					
D[3]					
D[4]					
C[3]					
C[2]					
C[4]					
C[2]					

Question 8 [20 Marks]

In this exercise, we will examine how replacement schemes affect miss rates. Assume a two-way set associative cache with four one-word blocks. Consider the following word address sequence: 0, 1, 2, 3, 4, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0.

(a) **[05 Marks]** Assuming an LRU replacement scheme, which accesses are hits?

Solution:

Block Address	Cache Index	Hit/ Miss	Evicted Block	Cache Block After Reference			
				Set 0	Set 0	Set 1	Set 1
0	0	Miss	-	Mem[0]			
1	1	Miss	-	Mem[0]		Mem[1]	
2	0	Miss	-	Mem[0]	Mem[2]	Mem[1]	
3	1	Miss	-	Mem[0]	Mem[2]	Mem[1]	Mem[3]
4	0	Miss	Mem[0]	Mem[4]	Mem[2]	Mem[1]	Mem[3]
5	1	Miss	Mem[1]	Mem[4]	Mem[2]	Mem[5]	Mem[3]
6	0	Miss	Mem[2]	Mem[4]	Mem[6]	Mem[5]	Mem[3]
7	1	Miss	Mem[3]	Mem[4]	Mem[6]	Mem[5]	Mem[7]
0	0	Miss	Mem[4]	Mem[0]	Mem[6]	Mem[5]	Mem[7]
1	1	Miss	Mem[5]	Mem[0]	Mem[6]	Mem[1]	Mem[7]
2	0	Miss	Mem[6]	Mem[0]	Mem[2]	Mem[1]	Mem[7]
3	1	Miss	Mem[7]	Mem[0]	Mem[2]	Mem[1]	Mem[3]
4	0	Miss	Mem[0]	Mem[4]	Mem[2]	Mem[1]	Mem[3]
5	1	Miss	Mem[1]	Mem[4]	Mem[2]	Mem[5]	Mem[3]
6	0	Miss	Mem[2]	Mem[4]	Mem[6]	Mem[5]	Mem[3]
7	1	Miss	Mem[3]	Mem[4]	Mem[6]	Mem[5]	Mem[7]
0	0	Miss	Mem[4]	Mem[0]	Mem[6]	Mem[5]	Mem[7]

Blue = new access, red = evicted block, black = previous ones.
As it can be clearly seen, there are no hits.

(b) **[05 Marks]** Most Recently Used (MRU) is a cache replacement scheme which removes the most recently used items first. A MRU scheme is good in situations in which the older an item is, the more likely it is to be accessed. Assuming an MRU (most recently used) replacement scheme, which accesses are hits?

Solution:

Block Address	**Cache Index**	**Hit/ Miss**	**Evicted Block**	**Cache Block After Reference**			
				Set 0	**Set 0**	**Set 1**	**Set 1**
0	0	Miss	-	Mem[0]			
1	1	Miss	-	Mem[0]		Mem[1]	
2	0	Miss	-	Mem[0]	Mem[2]	Mem[1]	
3	1	Miss	-	Mem[0]	Mem[2]	Mem[1]	Mem[3]
4	0	Miss	Mem[2]	Mem[0]	Mem[4]	Mem[1]	Mem[3]
5	1	Miss	Mem[3]	Mem[0]	Mem[4]	Mem[1]	Mem[5]
6	0	Miss	Mem[4]	Mem[0]	Mem[6]	Mem[1]	Mem[5]
7	1	Miss	Mem[5]	Mem[0]	Mem[6]	Mem[1]	Mem[7]
0	0	Hit	-	Mem[0]	Mem[6]	Mem[1]	Mem[7]
1	1	Hit	-	Mem[0]	Mem[6]	Mem[1]	Mem[7]
2	0	Miss	Mem[0]	Mem[2]	Mem[6]	Mem[1]	Mem[7]
3	1	Miss	Mem[1]	Mem[2]	Mem[6]	Mem[3]	Mem[7]
4	0	Miss	Mem[2]	Mem[4]	Mem[6]	Mem[3]	Mem[7]
5	1	Miss	Mem[3]	Mem[2]	Mem[4]	Mem[5]	Mem[7]
6	0	Hit	-	Mem[4]	Mem[6]	Mem[5]	Mem[7]
7	1	Hit	-	Mem[4]	Mem[6]	Mem[5]	Mem[7]
0	0	Miss	Mem[6]	Mem[4]	Mem[0]	Mem[5]	Mem[7]

The hit accesses are shown above. Blue = new access, Red = evicted, Black = previous ones, Green = Hit on the Block.

(c) **[05 Marks]** Describe an optimal replacement scheme for this sequence. Which accesses are hits using this policy?

Solution: MRU is the optimal replacement scheme for this sequence. The second 0, 1, 6, and 7 accesses are hits using this policy. LRU had no hits.

(d) **[05 Marks]** Describe why it is difficult to implement a cache replacement scheme that is optimal for all address sequences.

Solution: To implement a cache replacement scheme optimal for all address sequences, one would have to know the best block in each sequence to evict that will cause the fewest misses. However, each sequence is going to be unique, so each sequence will have a different block that should be evicted, and a cache can not know which block to evict in order to cause the fewest number of misses. This makes it difficult to implement such a scheme that is optimal for all address sequences.