Preliminary Report of Lab 6

CS224
Section No: 5
Spring 2021
Lab No: 6
Full Name/Bilkent ID: Esad İsmail Tök/21801679

Part 1 - Question 1

No.	Cache Size KB	N way cache	Word Size in bits	Block size (no. of words)	No. of Sets	Tag Size in bits	Index Size (Set No.) in bits	Word Block Offset Size in bits	Byte Offset Size in bits	Block Replacement Policy Needed (Yes/ No)
1	8	1	8	8	2^{10}	19	10	3	0	No
2	8	2	16	8	2^{8}	20	8	3	1	Yes
3	8	4	16	4	2^{8}	21	8	2	1	Yes
4	8	Full	16	4	2^{0}	29	0	2	1	Yes
9	32	1	16	2	2^{13}	17	13	1	1	No
10	32	2	16	2	2^{12}	18	12	1	1	Yes
11	32	4	8	8	2^{10}	19	10	3	0	Yes
12	32	Full	8	8	2^{0}	29	0	3	0	Yes

Table 1 (Table of Question 1)

Part 1 - Question 2

a)

- The first access to the memory address 0xA4 will be a compulsory miss. After the miss, a block of data will be fetched from memory to cache including the datas at the addresses: 0xA0 (not used), 0xA4, 0xA8, and 0xAC. Therefore all of the next accesses of those addresses will be hits.

Instruction	Iteration No.				
	1	2	3	4	5
lw $t1, 0xA4($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t2, 0xA8($0)	Hit	Hit	Hit	Hit	Hit
lw $t3, 0xAC($0)	Hit	Hit	Hit	Hit	Hit

Table 2 (Table of Question 2 - Part a)

b)

According to the question description, the capacity of the given cache memory is 8 words. The block size is 4 words. And this is a direct mapped cache memory.
Therefore in our address representation we have 2-bits byte offset, 2-bits block offset to select among the 4 words in a block, 1-bit set bit to select the set in the cache, and the most significant 27-bits are the Tag bits.

- Therefore in 1 set we have: 1-bit for the V (Valid), 27-bits for the Tag, and 4 times 32-bits for the data words.
- If we add the numbers: 1 + 27 + 4*32 = 156 bits
- In total we have 156-bits for one set in our cache memory.

- In one set we have 156-bits and in our cache we have 2 sets.
- If we do calculations: 156*2 = 312 bits
- So we have 312-bits total cache memory size.

c)

In this cache implementation we need:
1 AND gate,
We do not need any OR gate,
1 Equality comparator,
1 Multiplexer (4:1 MUX).

Part 1 - Question 3

a)

- The first accesses of the addresses 0xA4 and 0xA8 will be compulsory misses. Then when we try to access the address 0xAC, the cache will be full but since it is the first time that we access this address and since 0xAC is not previously removed from the cache because the cache was full, first access to the address 0xAC is also a compulsory miss. All the other accesses are capacity misses since they all replaced because of full capacity.

Instruction	Iteration No.				
	1	2	3	4	5
lw $t1, 0xA4($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t2, 0xA8($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t3, 0xAC($0)	Compulsory	Capacity	Capacity	Capacity	Capacity

Table 3 (Table of Question 3 - Part a)

b)

According to the question description, the capacity of the given cache memory is 2 words. The block size is 1 words. And this is a 2-way associative cache memory.
Therefore in our address representation we have 2-bits of byte offset and the most significant 30-bits are the Tag bits.

- The cache memory has only 1 set and it contains 2 ways (blocks): way0 and way1.
- In order to implement LRU policy we need to decide which way (block) is least recently used. To do this we can designate a *use bit (U)*, which is only 1-bit since we have only 2 blocks in the set. If the U = 0, then way0 is the least recently used block and the data in it is to be replaced. And if the U = 1, then way1 is the least recently used block and the data in it is to be replaced.
- Therefore, for a set only 1-bit is needed to implement the LRU policy.
- Since we have only 1 set in the cache memory, we again only need 1-bit in the entire cache memory to implement the LRU policy.

- In total we have only 1 set in the cache memory, and this set contains 2 ways (block).
In total the cache contains: 1-bit for the "U" bit to apply the LRU policy, 2 * 1-bit for the V (Valid) bit of the both blocks, 2 * 30-bits for the Tag bits of the both blocks, 2 * 32-bits for the word bits of the both blocks.

- If we do the calculations: 1 + (2 * 1) + (2 * 30) + (2 * 32) = 127 bits
- So we have 127-bits total cache memory size.

Part 1 - Question 3 (Continue)

c)

In this cache implementation we need:
2 AND gates,
1 OR gate,
2 Equality comparators,
1 Multiplexer (2:1 MUX)