Self-evaluation Exercise on Caches/memory system/TLB

Q1. Consider a fully associative cache following the LRU replacement scheme and consisting of only 8 words. Consider the following sequence of memory accesses (the numbers denote the word address):
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 22, 30, 21, 23, 31

Assume that we begin when the cache is empty. What are the contents of the cache after the end of the sequence of memory accesses?

Q2. Answer previous question assuming a FIFO replacement scheme.

Q3. What is the total size (in bytes) of a direct mapped cache with the following configuration in a 32 bit system? It has a 10 bit index, and a data-block size of 64 bytes. Each block has 1 valid bit and 1 dirty bit.

(You have to show the total size of **data + tag + valid bit + dirty bit** in the entire cache).

Q4. Consider a processor with 16-bit virtual and physical addresses. This processor uses a virtual to physical address translation scheme using a page table (even though both virtual and physical address spaces are of same size). The cache in the system has the following characteristics: 8 sets with 256B block size, 4-way associativity. Also, the cache is virtually-indexed, physically-tagged. It is given that binary representation of address 0xCDAB is 1100 1101 1010 1011.

a) Given that the page size is **512-byte**. In the diagram below, please mark with crosses (X), all the locations where **a virtual address** 0xCDAB can be stored.

Set Index	Way 0	Way 1	Way 2	Way 3
0				
1				
2				
3				
4				
5				
6				
7				

(b) Given that the page size is **512-byte**. In the diagram below, please mark with crosses (X), all the locations where **a physical address** 0xCDAB can be stored.

Set Index	Way 0	Way 1	Way 2	Way 3
0				
1				
2				
3				
4				
5				
6				
7				

c) Now, assume that the page size is **2048-byte**. In the diagram below, please mark with crosses (X), all the locations where **a physical address** 0xCDAB can be stored.

Set Index	Way 0	Way 1	Way 2	Way 3
0				
1				
2				
3				
4				
5				
6				
7				

Q5. Consider the data structures in two cases:

Case 1:

```
int RollNumber[SIZE];
int ExamMarks[SIZE];
```

Case 2:

```
struct StudentRecord {
```

```
int RollNumber_;
int ExamMarks_;
};
struct StudentRecord merged_array[SIZE];
```

Assume that a program reads the roll numbers of students from a file in the following format:

RollNumber Marks

101 45

102 56

....

Which of the data structures (case 1 or 2) will be more cache-friendly and why?

Q6. For each of these statements, write whether the statements are true or false. Also write one line explanation/reasoning. Note that in cache terminology, “cache line” and “cache block” are used synonymously.

Doubling the line size (while keeping total cache capacity same) halves the number of tags in the cache

Doubling the associativity (while keeping total cache capacity same) doubles the number of tags in the cache.

Doubling cache capacity of a direct-mapped cache usually reduces conflict misses.

Doubling cache capacity of a direct-mapped cache usually reduces compulsory misses

Doubling the line size usually reduces compulsory misses.