Computer Architecture
Assignment 9

Question 1:

Size of Cache = 1024 B = 2^{10} B
Size of Line = 64 B = 2^6 B
number of blocks in cache = 2^4 = 16

1) Direct Mapping:

In Direct mapping we use hash mapping using some hash functions. Simplest those is modulus num blocks in cache.
If we have 32 bit of memory Address we can split it into 2 parts 26-bit Tag and 6-bit Offset. Offset is to find where content is stored within line. Tag identifies which line/block it is.
We can further divide the tag error in 2 parts: last 4 bit to perform hashing and remaining to check what comes as a result of hashing has the same address whose content we want to retrieve.

Tag: 22 bits Index: 4 bits Offset: 6 bits

When we are using the modulus mapping function. We often do some tricks to ease our process. If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed.
i = j modulo m
where
i=cache line number
j= main memory block number
m=number of lines in the cache

If we want to read some memory, we calculate the hash value of the 4 bit index. And go to that corresponding hash value in the tag array and check if it is a hit or miss.

2) Fully Associative(FA) cache

This way of mapping gives us more flexibility as any block can go into any line of the cache. This means that the word id bits are used to identify which word in the block is needed, but the tag becomes all of the remaining bits. This enables the placement of any word at any place in the cache memory. This type of mapping is considered to be the most flexible mapping form.
Tag: 26 bits offset: 6 bits
Here we don't have any specific hash function, so search time is the number of blocks in cache. Which makes it very expensive considering time domain. We have to go through each and every line and check whether we have the same tag part of the memory address or not.
But it lets us keep the most frequent ones in cache. Which was one of the drawbacks of DIrect mapping.

3) 2 way set associative cache

Now we divide the cache into sets of two lines. So, now we have 8 sets(16 blocks / 2). Within the set it is direct mapping and sets are fully associative. Which tries to take advantage of both methods. Set associative addresses the problem of possible thrashing in the direct mapping method. It does this by saying that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a ***set***. Then a block in memory can map to any one of the lines of a specific set..Set-associative mapping allows that each word that is present in the cache can have two or more words in the main memory for the same index address.

Tag: 22 bits set index: 3 bits set offset: 1bit Offset: 6 bits

$m = v * k$
$i = j \bmod v$

where
i=cache set number
j=main memory block number
v=number of sets
m=number of lines in the cache number of sets
k=number of lines in each set

If we want to find something in cache we look for the index we go to and check within the set whether that tag is in that set or not. If it is there it is hit otherwise it is missed.
Sets are direct mapped, to search in sets we have to go through every element in that set. This way we reduced time complexity of FA and space problems of Direct mapping.