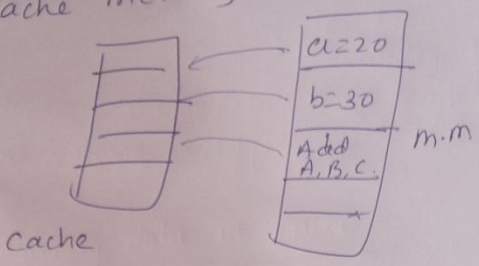
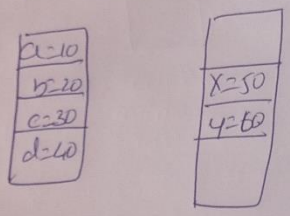
Cache mapping technique

CPU searches data in the cache,

1) Initially cache memory is empty, so the frequently accessed data has to be loaded into cache memory



2) ~~Cache~~ consider cache is full and new data has to be loaded into cache.



To load X or Y one of data item in the cache has to removed, to make a space for new data.

In the above two situations, one has to decide. which data block of main memory is to be loaded in which block of the cache?

Different mapping techniques will help to manage mapping of data from main memory to cache memory.

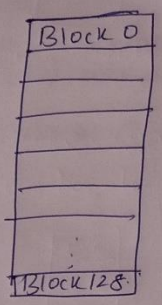
1) Direct mapped cache.

A simple way to determine cache locations in which to store memory blocks is the direct-mapping technique

In this

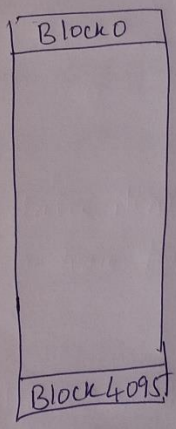
Block j of the m.m maps to → Block j % Total no of blocks in cache

Eg. consider a cache with 128 blocks of 16 words each



1 cache block can hold 16 words and we have a m.m with 64K words & no of bits in m.m is 16

i.e



Address
0000 0000 0000 0000
1111 1111 1111 1111

main memory totaly has 64 k words (total capacity)
K = 1024, each location capacity is 16 words value.

$$\frac{64 \times 1024 \text{ words}}{16}$$

= 4096 locations i.e from 0 to 4095

~~CPU generated~~

placement of a block in the cache is determined by m.m address

Given m.m address is now divided into 3 parts

1) word 2) Block 3) Tag bits

Tag	Block	word
5	7	4

word :- Total no of words in a location i.e 16

$2^4 = 16$ so 4 bits is for word

Block :- Totally we have 128 blocks in the cache i.e $2^7 = 128$ ∴ 7 bits for block.

Tag :- Remaining $(16 - (4+7)) = 5$ bits is for tag These bits are used to identify whether data is present in cache/not.

Eg :- Block 3 of m.m is stored in

$3 \% 128 = 3$ of the cache block.

Block 256 of the m.m is mapped to

$256 \% 128 = 0$ 0th block of the cache.

Block 1, 129, 257 are stored in cache block 1

$1 \% 128 = 1$

$129 \% 128 = 1$

$257 \% 128 = 1$

} 1st block of the cache.

Since more than one m.m block is mapped onto a given cache block position contention arises.

Associative mapping

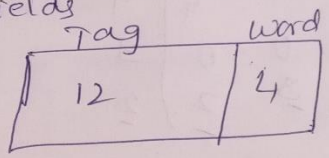
1) In this main memory block can be placed into any cache block position.

2) Space in the cache can be utilized more efficiently.

3) If the cache is full, new data to be loaded then to remove one of the cache block, least recently page replacement algorithm is used

4) cost of associative cache is higher than the cost of direct mapped cache.

5) main memory address is divided into two fields



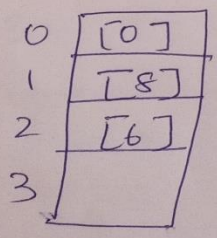
word :- 4 bits
since $2^4 = 16$ words

Tag :- Remaining bits are tag bit

Simple example

consider cpu has requested the data present in the main memory of 0, 8, 0, 6, 8

and we have four blocks in the cache



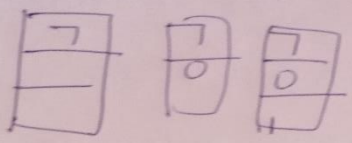
1) 0^{th} block of m.m is loaded in 0^{th} of cache
2) 8 of m.m => 1 of cache
3) 0 -> hit
4) 6 of m.m => 2^{nd} block of cache.
5) 8^{th} -> hit

2) Example :- working of LRU

7, 0, 1, 2, 0, 3 0 4 2 3 0 3 2.

m, m blocks

1) we have only 3 cache blocks, start loading



2) Now to load 2, LRU algorithm is used Just now 1 is referenced before that 0 & least reffered is 7

3) So 7 is removed and 2 is load.

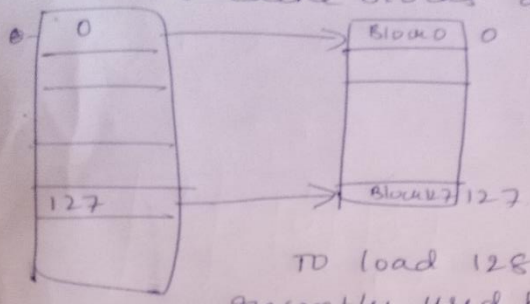
4) Next block is 0, already in cache so it is a hit,

3) Now to load 3 use LRU

Out 2, 0, 1 (~~2 is~~ 0 is just referenced, before that 2 and leastly referred is 1 so in the place of 1 3 is loaded

2	4	4	4	0
0	0	0	3	3
3	3	2	2	2.

For the previous example
i.e 128 cache blocks and 4096 m.m blocks

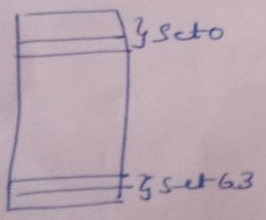


TO load 128, identify the least recently used block and that will be removed & 128 is stored in that position.

Set-associative cache

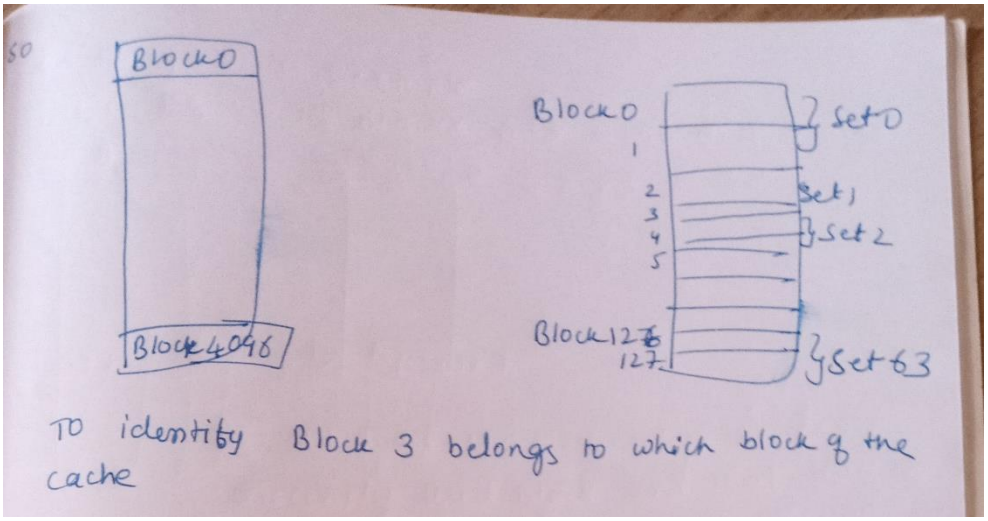
cache blocks are divided into number of sets

Eg



The sets can be n-way i.e if we group 2 blocks in one set, we call Two-way set associative

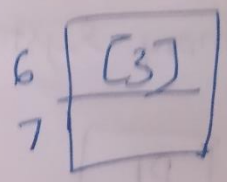
uses both direct mapping and Fully-associative method



To identify Block 3 belongs to which block of the cache

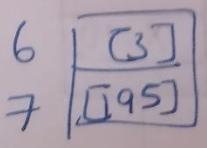
Set-associative

Block 3 % 64
= 3



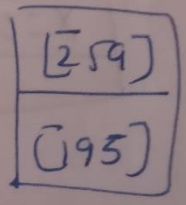
195 % 64
3

⇒



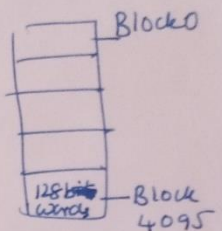
259 % 64
3

⇒



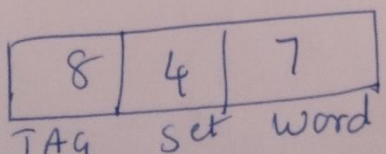
A block set-associative cache consists of a total of 64 blocks divided into 4-block sets. The main memory consists of 4096 blocks, each consisting of 128 words

a) How many bits are there in a main memory
b) How many bits are there in each of the TAG, SET and word fields?



a) Total no of blocks in m.m = 4096
Each block holds = 128 words
The capacity of m.m = 4096 × 128
= 524,288
2^{19} = 524,288
∴ 19 bits are there in m.m address

b) Out of 19 bits
TAG :- Remaining 8 bits is for tag
SET :- 64 blocks / 4 = 16 ⇒ 4 bits
word :- 128 words (2^7) so 7 bits



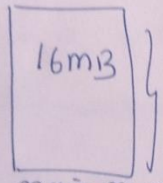
Problems on cache memory

1) A System has a main memory with 16 megabytes of addressable locations and a 32 kilobyte direct mapped cache with 8 bytes per block. The minimum addressable unit is a byte.

a) How many blocks are in the cache?

b) Show how the memory address is partitioned?

a) The main memory has 16 megabytes



$\Rightarrow$ 1 mB = 1024 kilobytes (kB)

$= 2^{10}$ kB

1 kB = 1024 bytes

1 mB = $2^{10}\ 2^{10}$ (Replace kB with 2^{10})

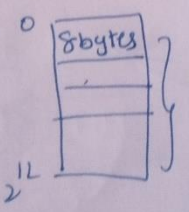
16 mB = $16\ 2^{10}\ 2^{10}$

$2^4\ 2^{10}\ 2^{10}$

$= 2^{24}$

So the total no of bits in main memory address = 24 bits

cache has



$\frac{32\ kB}{8}$

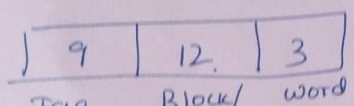
$\frac{32 \times 1024}{8}$

$2^2 \times 2^{10} = 2^{12}$

~~So 12 bits are required~~

2^{12} cache blocks

b) Direct map



out of 24 bit in address

Word: Since they have given in bytes, word ~~can~~ use bytes only

i.e 8 bytes = 2^3

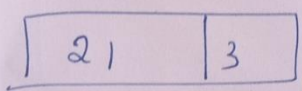
3 bit for word / byte

Block:- Totally 2^{12} blocks

so 12 bit for blocks

Tag = Remaining bit 9 is for tag field

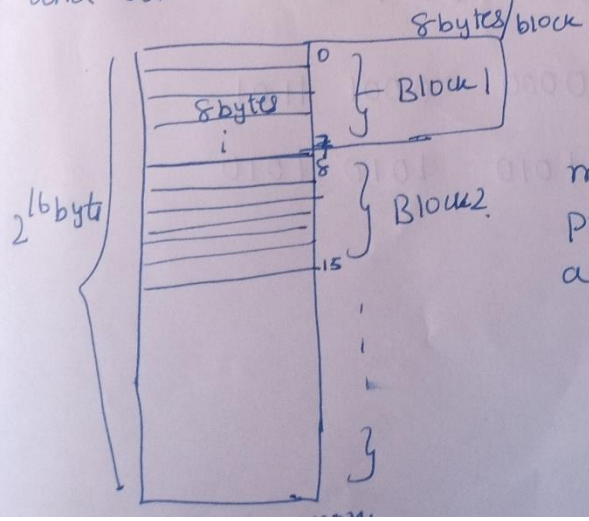
c) If it is a fully associative ~~ca~~ cache, then



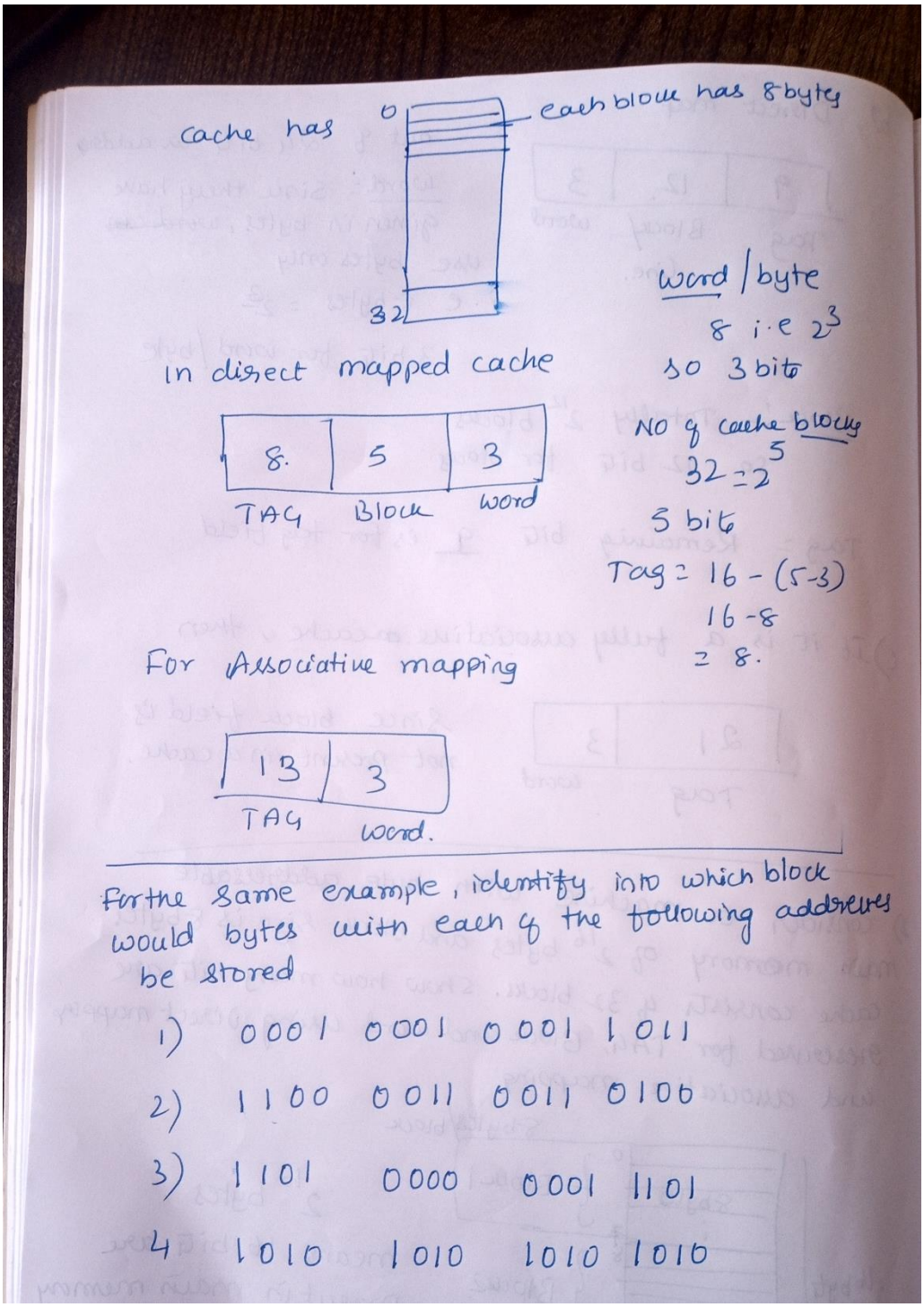
Since block field is not present in a cache.

---

2) consider a machine with byte addressable main memory of 2^{16} bytes and block size is 8 bytes cache consists of 32 blocks. Show how many bit are reserved for TAG, Block and word using Direct mapping and associative mapping.



2^{16} bytes means 16 bit are present in main memory address



Answer:

1) 3 2) 6

3) 3 4) 21