GATEWAY CLASSES

COA

Unit 4:MEMORY

ONE SHOT

Today's Target

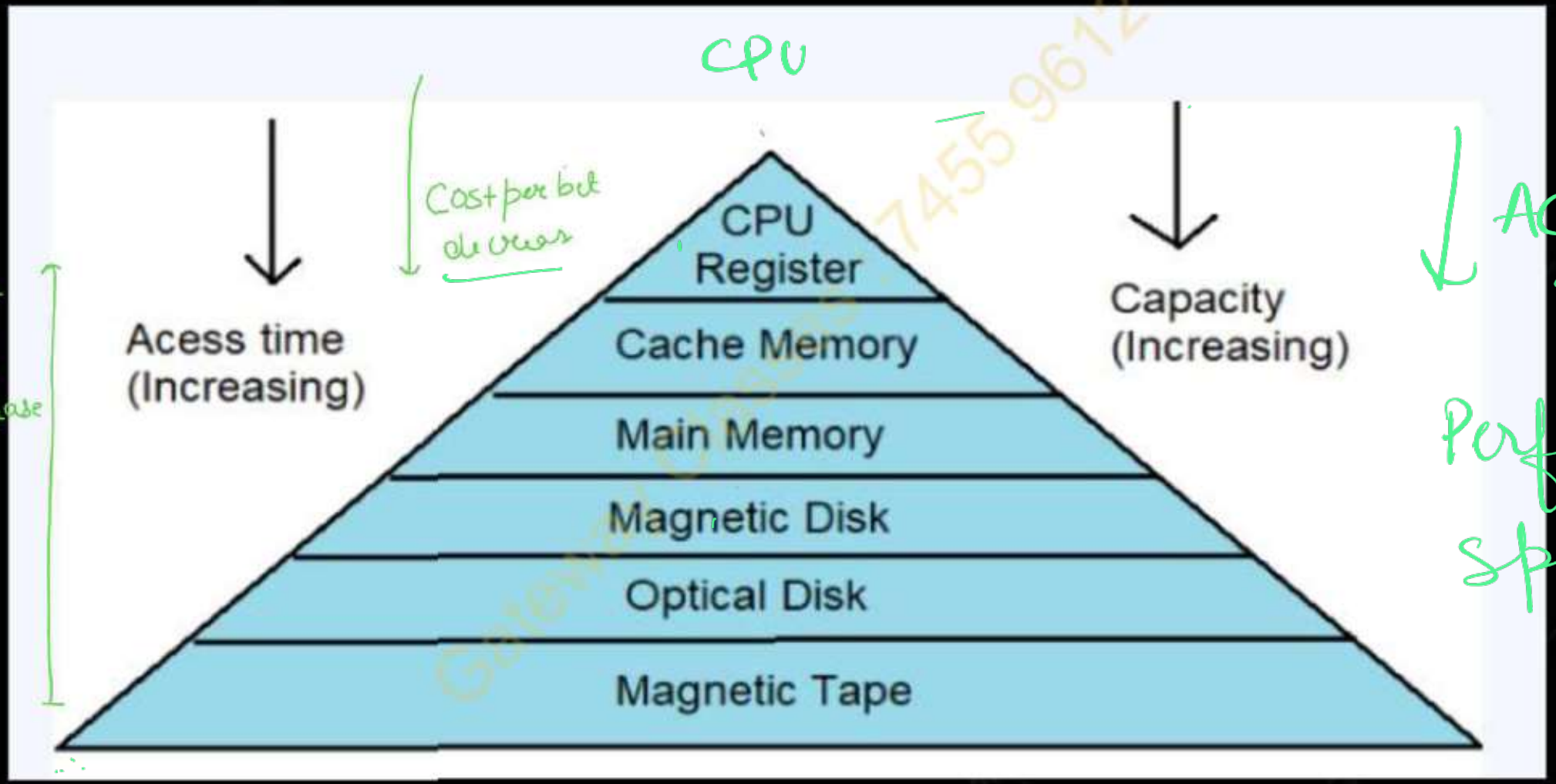
- MOST IMPORTANT TOPICS
- AKTU PYQs
- Result Oriented

By PRAGYA RAJVANSHI

B.Tech, M.Tech(C.S.E.)

PDF notes are available in the App (Link in Description)

Memory Hierarchy Design

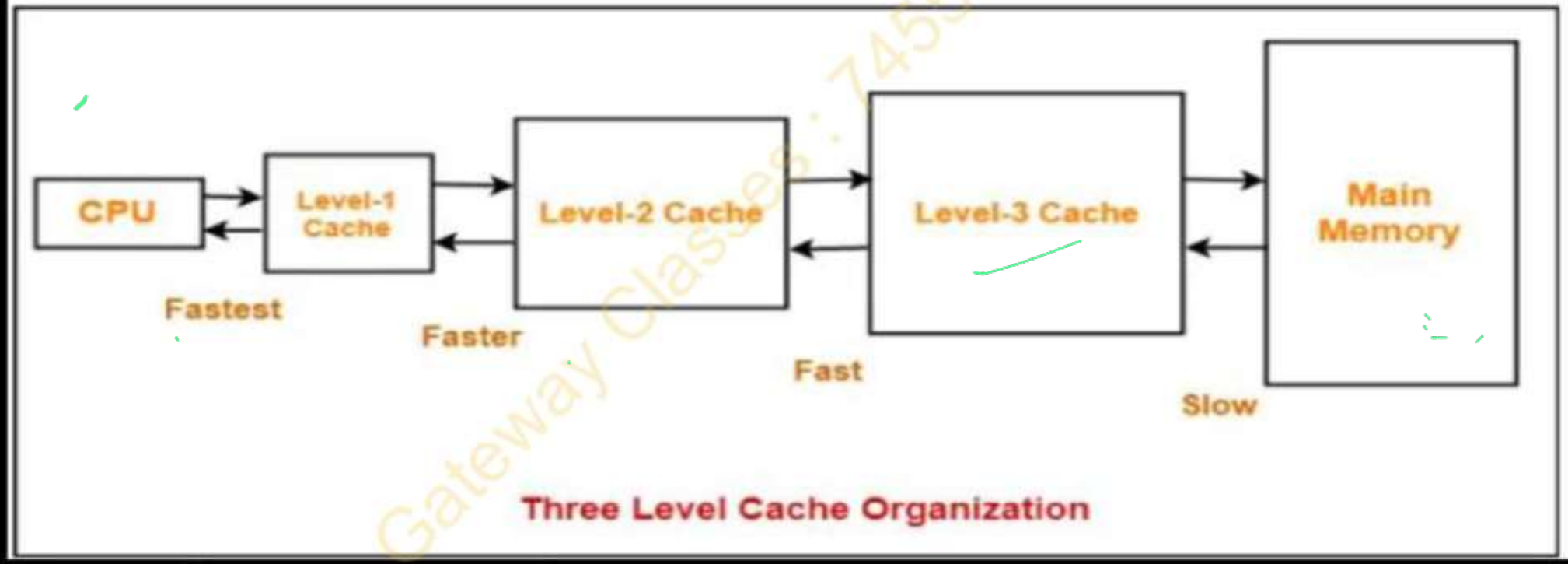


Multilevel Cache Organization

Cache - Small faster

MAT (Memory Access time)

Size (L1 Cache) < Size (L2 Cache) < Size (L3 Cache) < Size (Main Memory)



Three Level Cache Organization

CACHE PERFORMANCE

Cache Hit

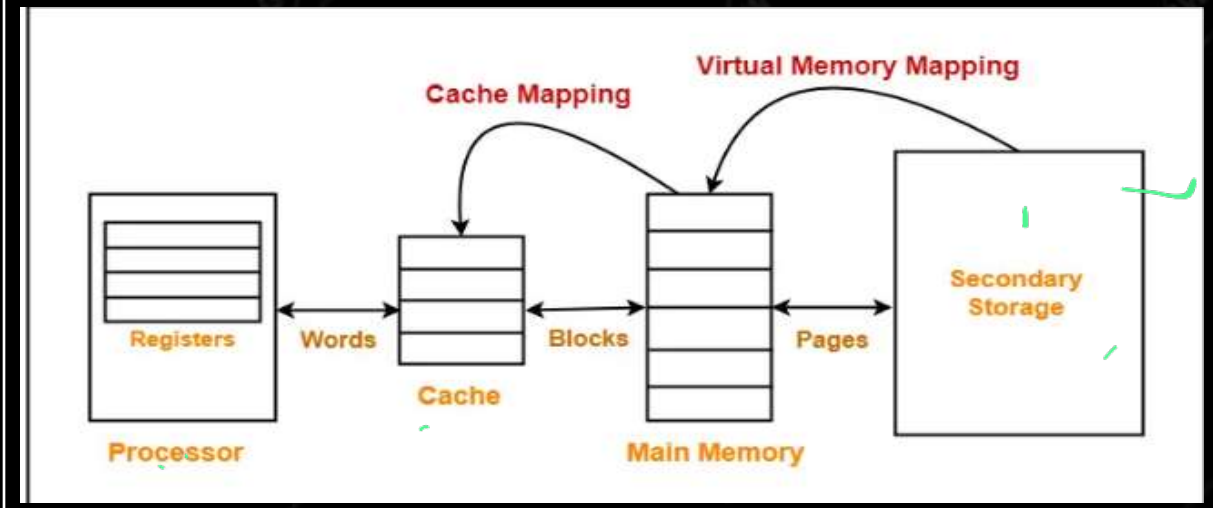
- When the processor needs to read or write a location in the main memory, it first checks for a corresponding entry in the cache.
- If the processor finds that the memory location is in the cache, a Cache Hit has occurred and data is read / Write from the cache.
- If the processor does not find the memory location in the cache, a **cache miss** has occurred. For a cache miss, the cache allocates a new entry and copies in data from the main memory, then the request is fulfilled from the contents of the cache.
- The performance of cache memory is frequently measured in terms of a quantity called **Hit ratio.**

Cache Mapping-

Cache mapping is a technique by which the contents of main memory are brought into the cache memory.

Cache Mapping Techniques-

- Direct Mapping
- Fully Associative Mapping
- K-way Set Associative Mapping



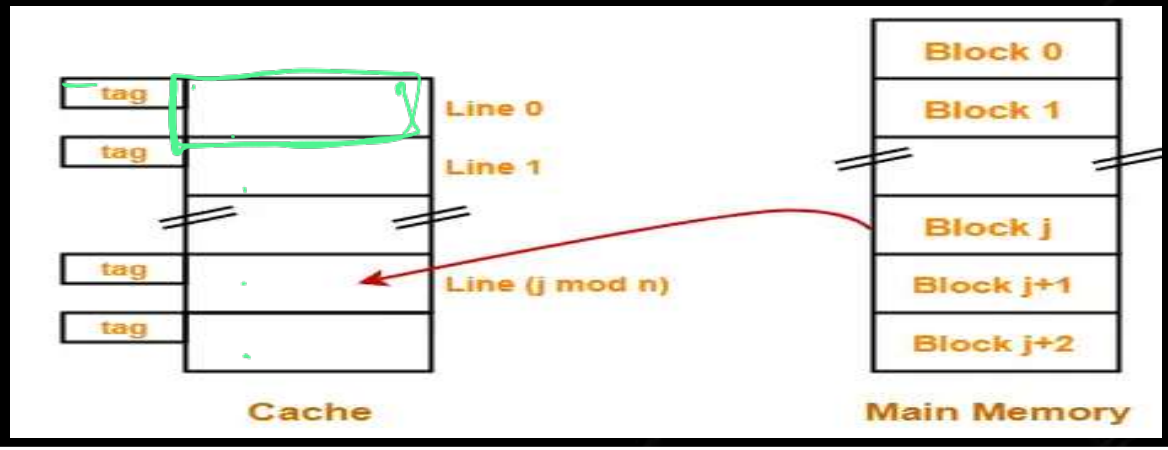
Direct Mapping-

In direct mapping, A particular block of main memory can map only to a particular line of the cache.

The line number of cache to which a particular block can map is given by

Cache line number
= (Main Memory Block Address) Modulo (Number of lines in Cache)

- Consider cache memory is divided into 'n' number of lines.Then, block 'j' of main memory can map to line number (j mod n) only of the cache.
- There is no need of any replacement algorithm. This is because a main memory block can map only to a particular line of the cache. Thus, the new incoming block will always replace the existing block (if any) in that particular line.

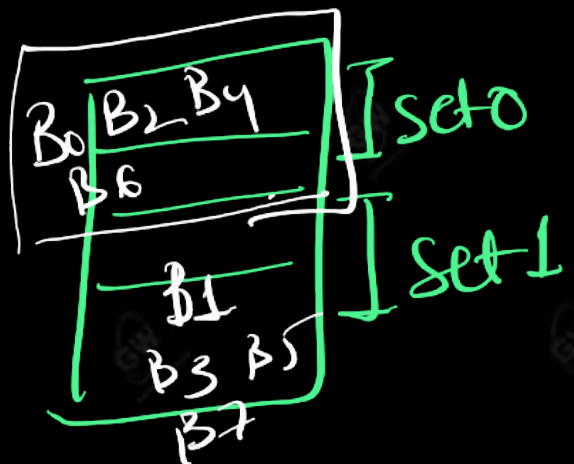


K-way set Associative Mapping

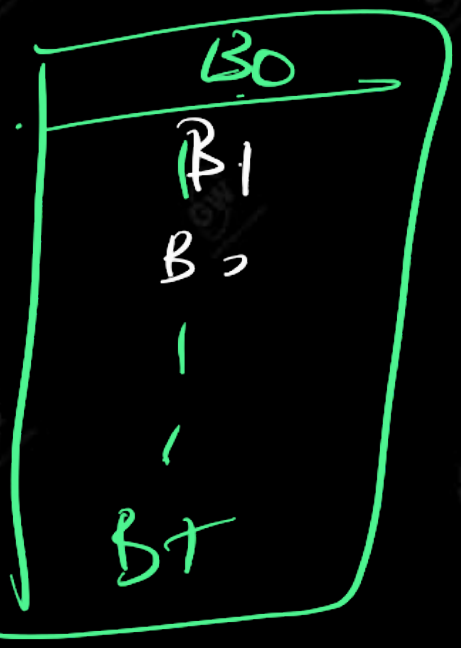
Derived from fully

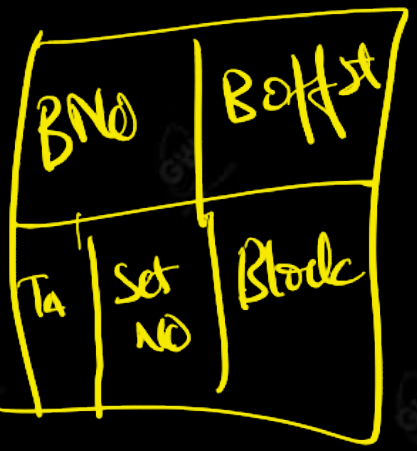
K = No of lines in a set

K = 2



$\frac{4}{2} = 2$ set





J mod n

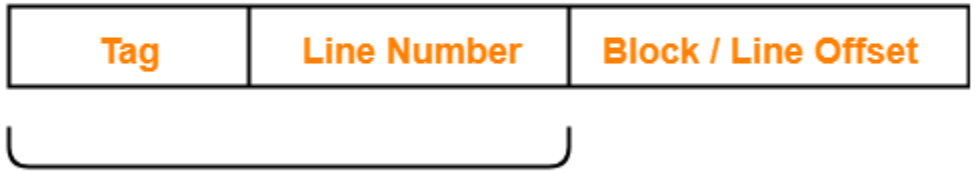
↳ set number

↓

Block

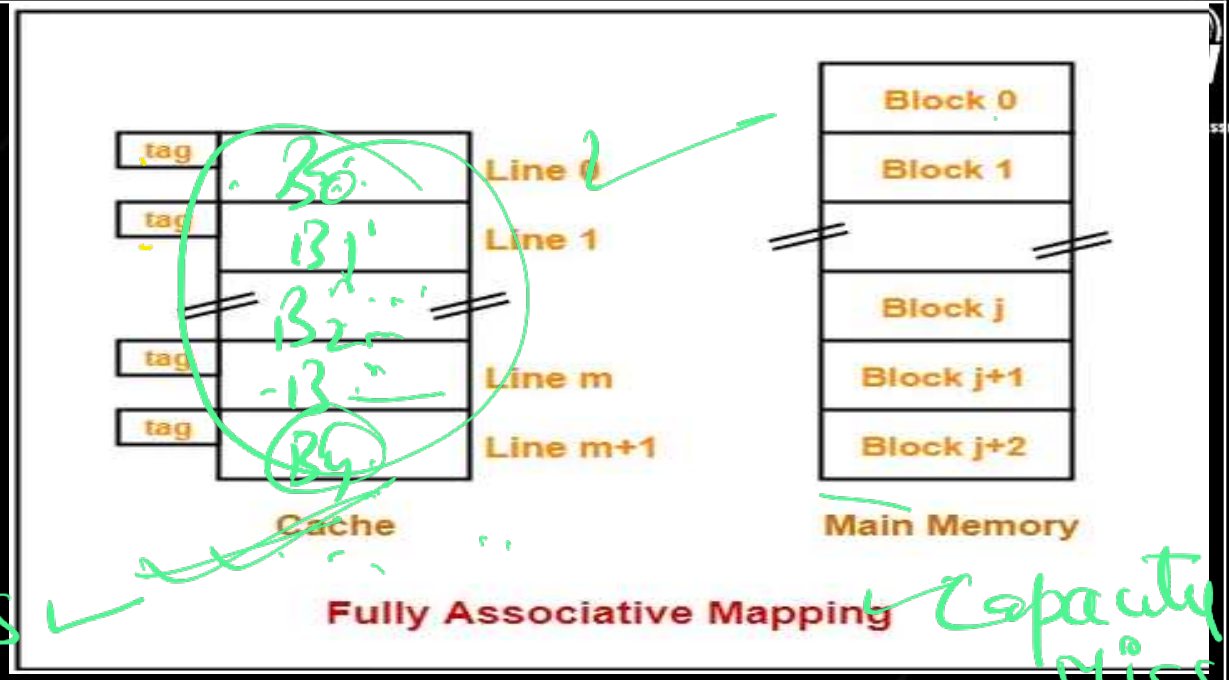
Division of Physical Address-

In direct mapping, the physical address is divided as-



2. Fully Associative Mapping-

- A block of main memory can map to any line of the cache that is freely available at that moment.
- This makes fully associative mapping more flexible than direct mapping.
- All the lines of cache are freely available.
- Thus, any block of main memory can map to any line of the cache. Had all the cache lines been occupied, then one of the existing blocks will have to be replaced.



Fully Associative Mapping

- A replacement algorithm is required.
- Replacement algorithm suggests the block to be replaced if all the cache lines are occupied.
- Thus, replacement algorithm like FCFS Algorithm, LRU

Division of Physical Address in Fully Associative Mapping

K-way Set Associative Mapping-

- Cache lines are grouped into sets where each set contains k number of lines.
- A particular block of main memory can map to only one particular set of the cache.
- However, within that set, the memory block can map any cache line that is freely available.
- The set of the cache to which a particular block of the main memory can map is given by-

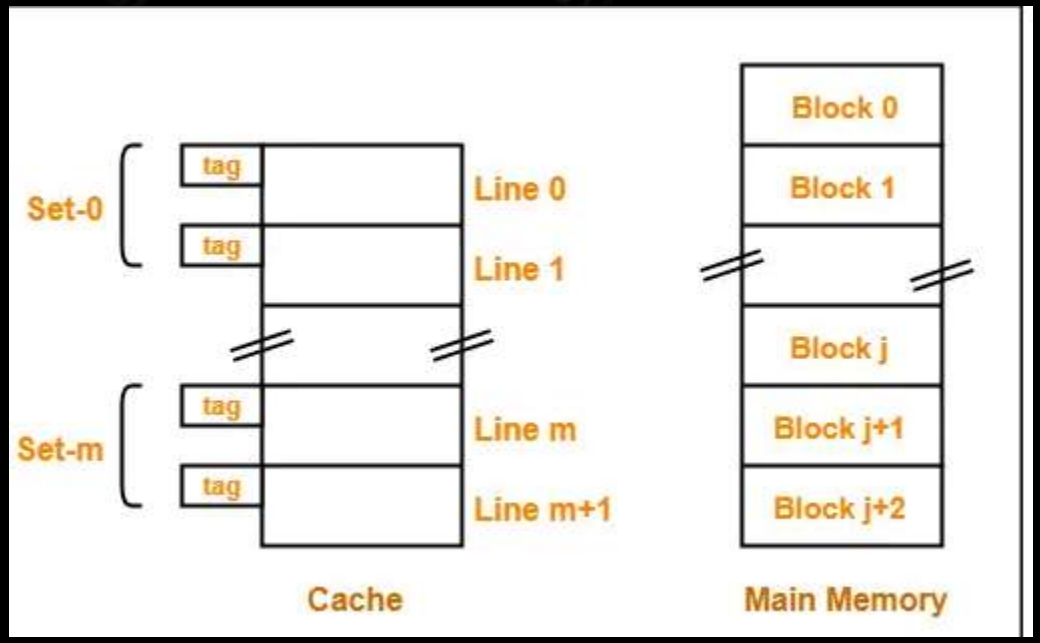
Cache set number
= (Main Memory Block Address) Modulo (Number of sets in Cache)

k = 2 suggests that each set contains two cache lines.

Since cache contains 6 lines, so number of sets in the cache = 6 / 2 = 3 sets.Block ‘j’ of main memory can map to set number (j mod 3) only of the cache

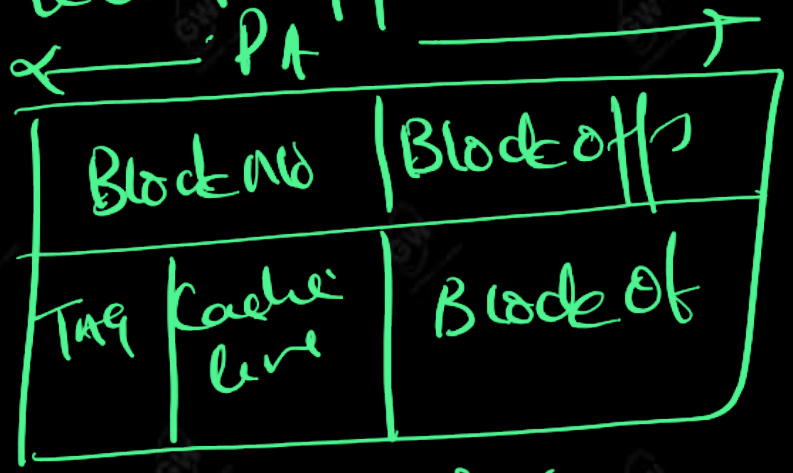
.Within that set, block ‘j’ can map to any cache line that is freely available at that moment.

If all the cache lines are occupied, then one of the existing blocks will have to be replaced



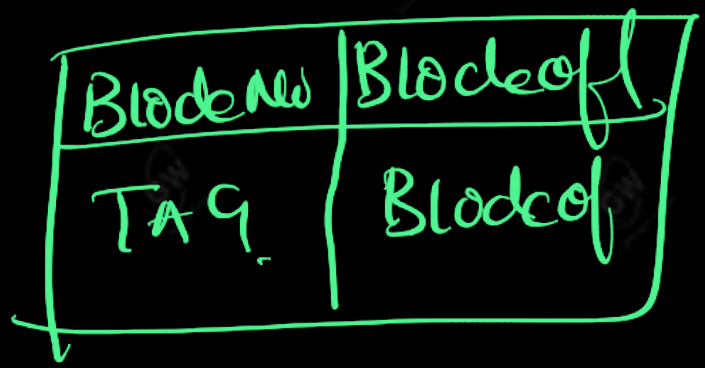
Division of Physical Address in K-way Set Associative Mapping

Direct mapping



→ conflict miss
→ compulsory
k mod m → No of cache line
↑
Block No
→ capacity miss

full Associative



→ conflict miss
→ compulsory miss
→ capacity miss

NUMERICALS

Problem-01

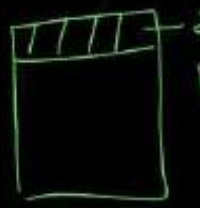
Consider a direct mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB.

Find-

Number of bits in tag

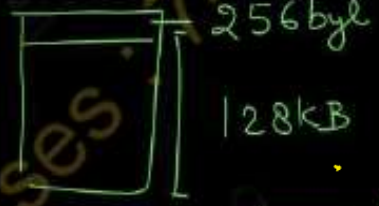
Tag directory size

Cache Memory
Cache Size = 16KB
Block Size = 256 byte



Cache Memory

Main memory
Size = 128kB
Block Size = 256 byte



NOTE
Size Block Size Main Memory = Cache Memory
Size Block | Cache line

① Physical Address of Main Memory
Size MM = 128kB
$2^7 * K * B$
$2^7 * 2^{10}$ Byte
2^{17} Byte

NOTE
$2^{10} = K$

(17)
Physical Address

Block Number (9)	Block offset (8)
TAG (3)	Cache line (6)	Block offset (8)

Cache

② Size of block = 256
2^8 = 8 bit needed

2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024

NOTE= Block offset/word = Number of bits used to represent a word

Cache Size = 16KB
1 block size = 256

No of Cache line = $\frac{16KB}{256B}$ $\frac{2^4 * 2^{10} B}{2^8 * B}$ $= 2^6$ lines

6 bit need to represent 2^6 lines

No of TAG = Physical Address − (No of bit in cache line + No of bit in Block offset)

$17 - (6+8)$

3 bit

TAG Directory = No of bit in tag * No of cache line

$3bit * 2^6$

$\frac{3*2^6}{8}$ byte $= \frac{3*2^6}{2^3}$ $3*2^3 = 3*8 = 24$ byte

NUMERICALS

Problem no-2:

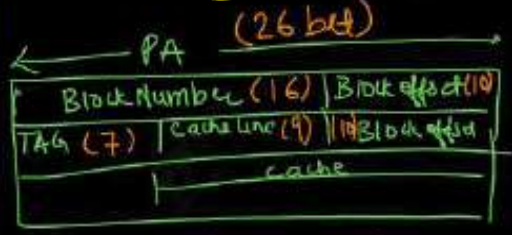
Consider a direct mapped cache of size 512 KB with block size 1 KB. There are 7 bits in the tag. Find-

1. Size of main memory
2. Tag directory size

Given
Cache size = 512 KB
block size = 1KB
TAG Bits = 7 bit

Block Size = 1kB
$1 * 2^{10}$ Byte
2^{10} Byte

No of bit to Represent a Word in block = 10 bits



2^{16}

(*) Number of cache line = Cache size / block size

$\frac{512\ KB}{1\ KB} = 512$ lines

2^9 lines

Number bit Needed to Represent 2^9 line = 9 bits

(*) Block number bit = No of bits in tag + No of bit need to Represent a Cache line
7 + 9 = 16 bit

(*) Number bit required to Represent a PA
bit in Block number + Bit in Block offset
16 + 10 = 26 bit

Size of Main memory
2^{26} Byte
$2^6 * 2^{20}$ Byte
64 MB

$2^{10} - K$
$2^{20} - M$
$2^{30} - G$

$2^0 \rightarrow 1$	
$2^1 \rightarrow 2$	
$2^2 \rightarrow 4$	
$2^3 \rightarrow 8$	
$2^4 \rightarrow 16$	
$2^5 \rightarrow 32$	
$2^6 \rightarrow 64$	
$2^7 \rightarrow 128$	
$2^8 \rightarrow 256$	
$2^9 \rightarrow 512$	
$2^{10} \rightarrow 1024$	

Size of TAG Directory
= No of bit in tag * Cache line
$7 * 2^9$ bit

$\frac{7 * 2^9}{8}$

$\frac{7 * 2^9}{2^3}$

$7 * 2^6$ byte
$7 * 64$ byte
448 byte

① Number of bits used to Represent a block = 16 bit

② Number of bits used to Represent a word in memory (Cache/Main) = 10 bits

③ Number of bits used to Represent a TAG = 7 bits

④ Number of bits used to Reprise No of cache line = 9 bit

⑤ Number bits need to Represent an address in cache = 9+10 = 19 (cache line + Block offset)

⑥ Number of bits needed to Represent an address (PA) in Main Memory = 26 bits

NUMERICALS

Problem-04:

Consider a direct mapped cache of size 32 KB with block size 32 bytes. The CPU generates 32 bit addresses.

The number of bits needed for cache indexing and the number of tag bits are respectively-

- 10, 17
- 10, 22
- 15, 17
- 5, 17

Given cache size = 32 KB

Block size = 32 byte

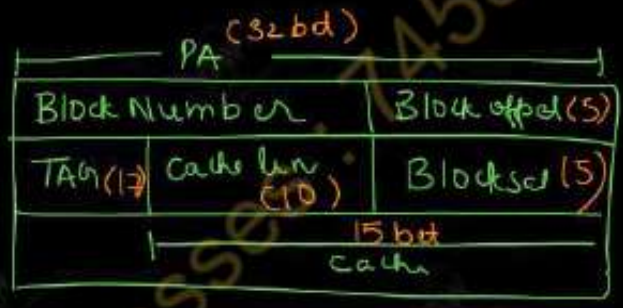
PA = 32 bit

No of Cache index = No of bit Cache line

Block size = 32 byte = 2^5

No of bit needed to Represent a Word in memory = 5 bits

Cache size = 32 KB = $2^5 \times 2^{10}$ Byte = 2^{15} byte



Number of bit needed to Represent an address in cache = 15 bit

bit to Represent a Cache line = No of bit to Represent a cache address − No of bit used to Represent a Word in memory

15 − 5 = 10 bit

Number of bit needed for Cache indexing = 10 bit

Number bit for TAG = bit to PA − bit to Represent Cache address

32 − 15 = 17

2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024

NUMERICALS

Q.1 Consider a fully associative mapped cache of size 8KB with block size 256 bytes. The size of main memory is 256KB Find-

- Number of bits in tag = 10 bits
- Tag directory size = $\frac{\text{TAG Bit} \times \text{Cache line}}{10 \times 2^5 \text{ bit}}$

Given

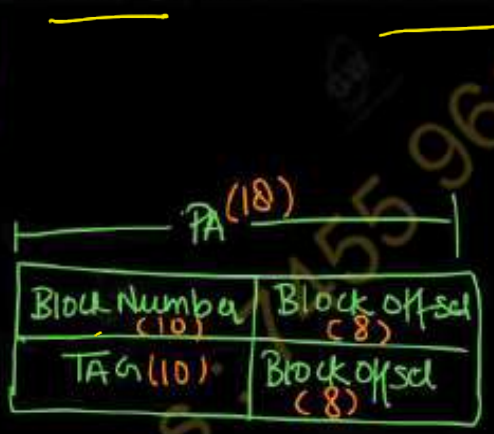
Cache Size = 8 kB

Block Size = 256 byte

Main Memory Size = 256kB

(1) Number of bits Required to Represent a Word in Memory = 8 bit

Block Size = $\frac{256 \text{ byte}}{2^8}$



Main Memory Size = 256kB

$2^8 \times 2^{10}$ B

2^{10} B

Number of bit to Represent PA = 18 bits

Number of block in main memory =

$\frac{\text{Size of Main Memory}}{\text{Block Size}}$

$\frac{256\text{kB}}{256\text{B}} = \frac{1\text{k}}{2^{10}}$ block

Number bit to Represent a block = 10 bit

Number of TAG Bit = No of bit of Block number = 10 bit

Number of cache line = $\frac{\text{Size of Cache}}{\text{Size of Block}}$

$\frac{8\text{kB}}{256\text{B}}$ $\frac{2^3 \times 2^{10}\text{ B}}{2^8\text{ B}} = 2^5$ Cache line

$2^0 \to 1$
$2^1 \to 2$
$2^2 \to 4$
$2^3 \to 8$
$2^4 \to 16$
$2^5 \to 32$
$2^6 \to 64$
$2^7 \to 128$
$2^8 \to 256$
$2^9 \to 512$
$2^{10} \to 1024$

1 byte = 8 bit

$\frac{1}{8}$ byte = 1 bit

TAG Directory = 10×2^5 bit

$\frac{10 \times 2^5}{8}$

$\frac{10 \times 2^{\cancel{5}^{2}}}{\cancel{2^3}} = 10 \times 2^2$

$10 \times 4 = 40$ Byte

Locality of reference

- Locality of reference refers to a phenomenon in which a computer program tends to access same set of memory locations for a particular time period. In other words, **Locality of Reference** refers to the tendency of the computer program to access instructions whose addresses are near one another. The property of locality of reference is mainly shown by loops and subroutine calls in a program
- In case of loops in program control processing unit repeatedly refers to the set of instructions that constitute the loop.
- In case of subroutine calls, every time the set of instructions are fetched from memory.
- References to data items also get localized that means same data item is referenced again and again.

Difference between

Spatial locality	Temporal locality
In Spatial Locality, nearby instructions to recently executed instruction are likely to be executed soon.	In Temporal Locality, a recently executed instruction is likely to be executed again very soon.
It is also known as locality in space.	It is also known as locality in time.
It only refers to data item which are closed together in memory.	It repeatedly refers to same data in short time span.
Each time new data comes into execution.	Each time same useful data comes into execution.
Example : Data elements accessed in array (where each time different (or just next) element is being accessing).	Example : Data elements accessed in loops (where same data elements are accessed multiple times).

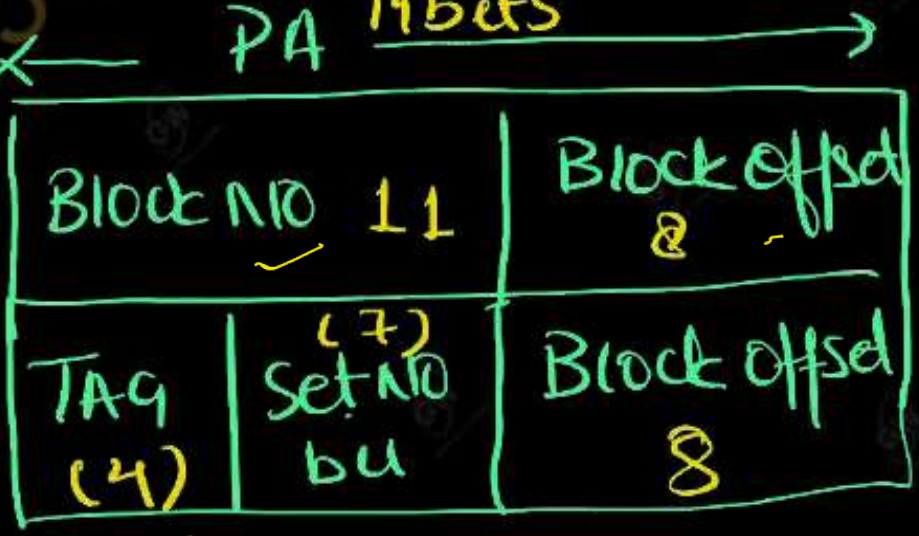
NUMERICALS

Problem-01:

Consider a 2-way set associative mapped cache of size 64 KB with block size 256 bytes. The size of main memory is 512 KB. Find-

- Number of bits in tag
- Tag directory size

Given 2-way set Associative
Cache Size = 64 KB
Block Size = 256 byte
Main Memory Size = 512 KB



Main Memory Size = 512KB

$2^9 \times 2^{10} B = 2^{19} B$

Number of bits in physical Address = 19 bit

Cache line = $\frac{\text{Size of cache}}{\text{Block size}}$

$\frac{64kB}{256B} = \frac{2^6 \times 2^{10} B}{2^8 B}$

$2^8 = 256$ Cache line

Number of set = Cache line / Number of cache line in a set

$\frac{2^8}{2} = 2^7$ set

Number of bits to Represent a set = 7

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$

Block size = 256 byte
2^8

Number of bits to Represent a word in Block : 8 b

Number of bit Needed to Represent a block = PA bit − Blockoffset bit

19 − 8 = 11 bit

*

PA bit = Block No bit + Block offset bit

Block Number bit = TAG Bit + Set No bit

11 = TAG Bit + 7

TAG Bit = 11 − 7 = 4 bit

TAG Directory size = Cache line × TAG Bit

$2^8 \times 4$ bit

$\frac{2^8 \times 4}{8}$ byte

$\frac{2^8 \times 2^2}{2^3} = 2^7$ byte

128 byte

TAG Directory size = 128 byte

NUMERICAL 2

➢A Digital computer has a memory unit of 64kx16 and cache memory of 1k words .The cache use direct mapping with a block size of 4 words(AKTUI 2021-22)

1. how many bits are in the tag, index , block and I word filed of the address format

2. how many bits are there in each word in cache and how they are individual into function include a valid function

3. how many blocks can the cache accommodate.

6 8 14 2

Memory unit = 64 K x 16 → No of bit bee words

↓

Number of words

Index = Cache line bi

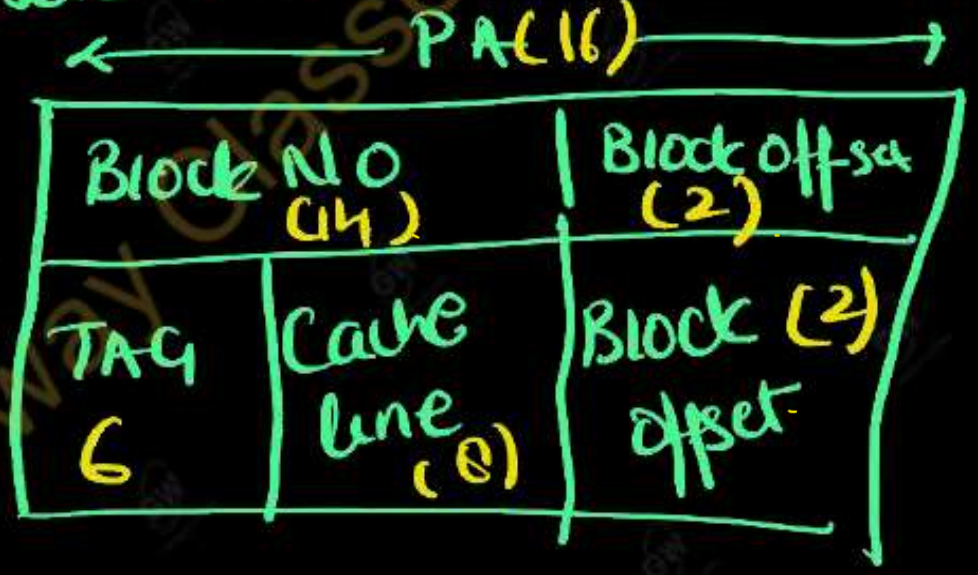
Main Memory size =
64K words
$2^6 \times 2^{10} = 2^{16}$ words

Cache Memory = 1K words
2^{10} words

Block size = 4 words $= 2^2$ words

Direct Mapping

One word = 16 bit
a data size = One word
16 bit =
2 byte



$2^0 \rightarrow 1$
$2^1 \rightarrow 2$
$2^2 \rightarrow 4$
$2^3 \rightarrow 8$
$2^4 \rightarrow 16$
$2^5 \rightarrow 32$
$2^6 \rightarrow 64$
$2^7 \rightarrow 128$
$2^8 \rightarrow 256$
$2^9 \rightarrow 512$
$2^{10} \rightarrow 1024$

Block Size = 2^2 words

Number of bit needed to Represent a word in Block | Block offset |

Word field = 2 bit

→ the Block Size of Main Memory = the size of Cache line | Block

→ Same bit for both Block offset of Main & Cache Memory

Main Memory Size

2^{16} words

Physical Address bit = 16 bit

Number of bit needed to Represent a block in memory

= PA bit − Block offset bit

16 − 2 = 14 bit

Number of block in main memory = 2^{14} blocks

$$\frac{\text{Main memory size}}{\text{Block size}} = \frac{2^{16} \text{ words}}{2^2 \text{ word}} = 2^{14} \text{ blocks}$$

$$\text{Cache line} = \frac{\text{Cache Size}}{\text{Block size}} = \frac{2^{10} \text{ words}}{4 \text{ word}}$$

$$\frac{2^{10} \text{ word}}{2^2 \text{ word}} = 2^8 \text{ Cach line}$$

Number of bit needed to Represent a cache line = 8 bit

Block Number bit = TAG Bit + Cache line bit

$14 = \text{TAG Bit} + 8$

$\text{TAG Bit} = 14 - 8 = 6$

(1)

TAG Bit → 6 bit

Block bit → 14 bit

Index / cache line bit → 8 bit

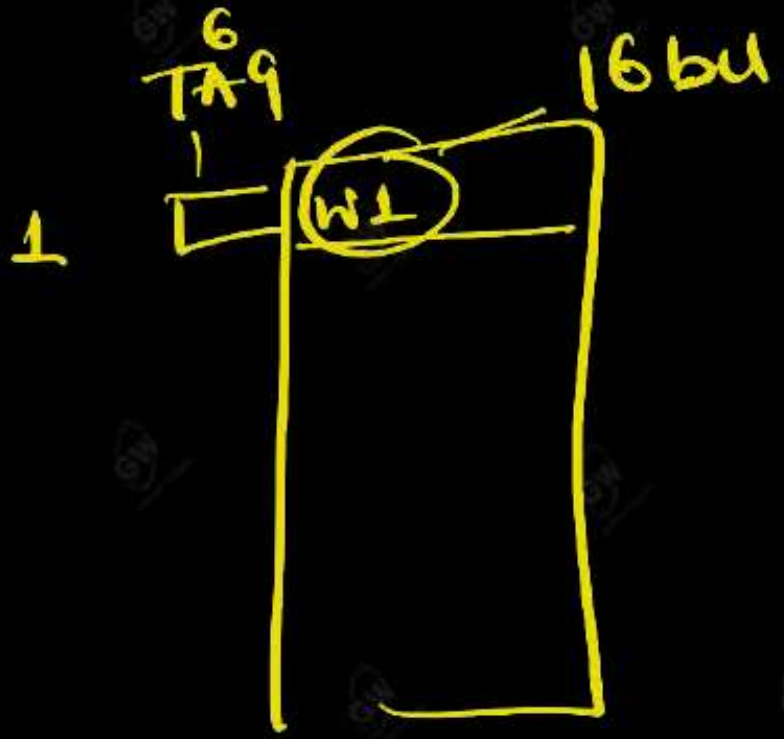
Word field / Block offset → 2 bit

W1

16 bit (data)

(1) bit valid bit

6 TAG Bit



(ii) total bit

Data bit + valid bit + TAG Bit

16 + 1 + 6 = 23 bit

© How many a Cache Memory can Accomodate

Cache Memory size = 1k words

Block size = 4 words

Number of block = $\frac{\text{Cache Memory size}}{\text{Block size}}$

$\frac{1k\ words}{4\ words} = \frac{2^{10}\ words}{2^2\ words}$

2^8

256 blocks

NUMERICAL 3

- a two way set associative cache memory uses block of 4 words .The cache can accommodate a total of 2048 words from memory .the main memory size is 128k x32

1 Formulate the all pertinent information required to construct the cache memory

2What is the size of cache memory (AKTU 2018-19)

TAG

2-way set Associative

Block size = 4 words { Main Memory & cache line

Cache size = 2048

Main Memory size = 128k x 32

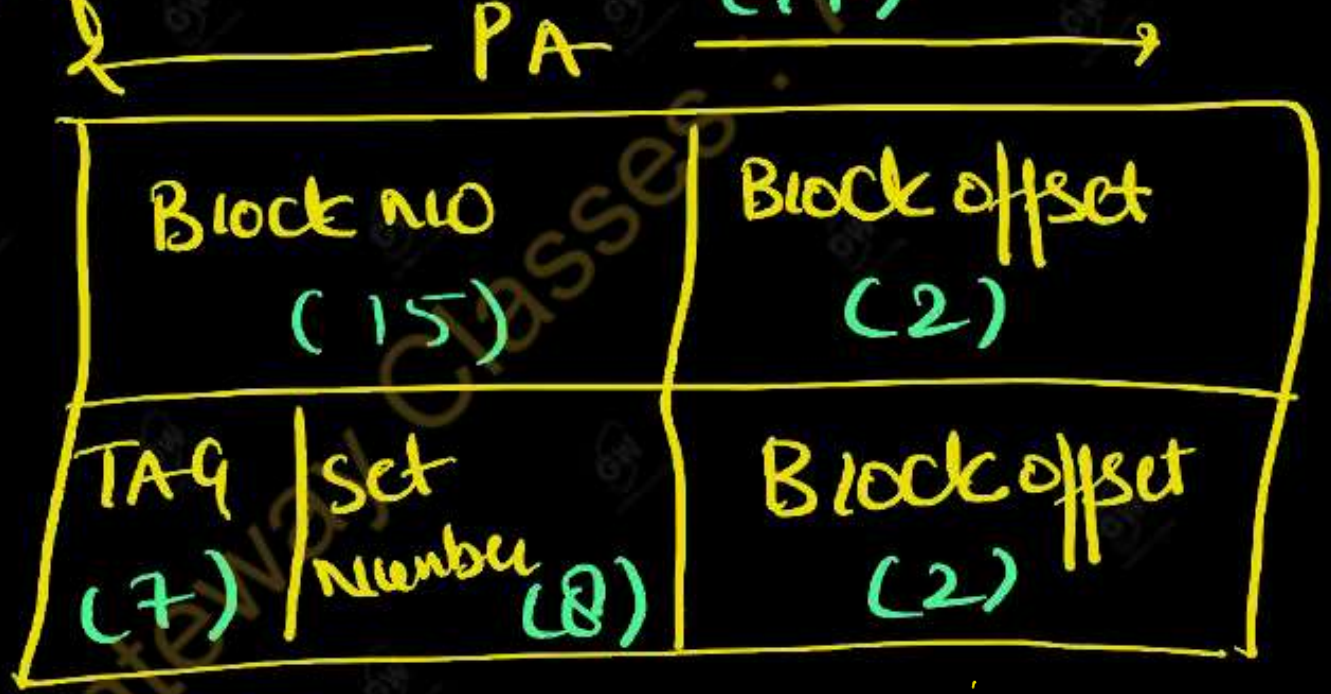
↓ words → No of bit per words

Main Memory size = 128K words

$2^7 \times 2^{10}$ words

2^{17} words

Number of bit needed for PA = 17 bits



Block size = 4 words = 2^2 words

Number of bit needed to Represent a words in memory | block | wordfield | Block offset = 2 bits

→ Number of bit needed to Represent a block in memory

PA-bit − Block offset bit

17 − 2 = 15 bit

Number of blocks in Main Memory = 2^{15} blocks

Cache line = $\frac{\text{Size of Cache Memory}}{\text{Block Size}} = \frac{2048}{4}$

2^9 ← 512 Cache line

Cache line

Number of set = $\frac{\text{Cache line}}{\text{Number of line in set}}$

$\frac{2^9}{2^2} = 2^8$ set

Number of bit needed to Represent a set

Block No bit = TAG Bit + Set No bit (Index)

15 = TAG Bit + 8

TAG Bit = 15 - 8 = 7

All the pretinet info needed to construct cache memory

TAG Bit → 7

Set No bit → 8

Block offset / word field = 2 bit

Cache line = 2^9

Cache set Number = 2^8

(11) Size of Cache Memory = 2048 words

2048×32 bit

$2^{11} \times 2^5$ bit

$2^6 \times 2^{10}$ bit

64 k bit

$\frac{64 \text{ k bit}}{8}$ ← 8

8 k byte → size Cache Memorys

Cache replacement algorithm

- When a block of main memory need to brought in while all the cache memory block are occupied one need to be replaced this is known as cache replacement algorthim
- Optimal replacement algorthim
- LRU replacement algorthim
- FIFO replacement algorthim
- REFERENCE STRING 3,2,1,3,4,1,6,2,4,3,4,2,1,4,5,2,1,3,4

3 FRAMES

Page faullt = 13

NUMERICALS

Miss Ratio = 13/19

- FIFO First in first Out

hit Ratio = cacu hit / Total = 6/19

Reference →	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5	2	1	3	4
F3			1	1	1	1	1	2	2	2	2	2	1	1	1	1	1	3	3
F2		2	2	2	2	2	6	6	6	6	4	4	4	4	4	2	2	2	2
F1	3	3	3	3	4	4	4	4	4	3	3	3	3	3	5	5	5	5	4
	X	X	X	✓	X	✓	X	X	✓	X	X	✓	X	✓	X	X	✓	X	X

X → Miss ✓ - Hit.

LRU → Least Recently used NUMERICALS Page fault = 14

Replace past a page which is not used for a long time en

	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5	2	1	3	4
F3			1	1	1	1	1	1	4	4	4	4	4	4	4	4	1	1	1
F2		2	2	2	4	4	4	2	2	2	2	2	2	2	5	5	5	3	3
F1	3	3	3	3	3	3	6	6	6	3	3	3	1	1	1	2	2	2	4
	x	x	x	✓	x	✓	x	x	x	x	✓	✓	x	✓	x	x	x	x	x

Miss Ratio = 14/19 Hit Ratio = 5/19

Optimal Page Replacement Algorithm NUMERICALS

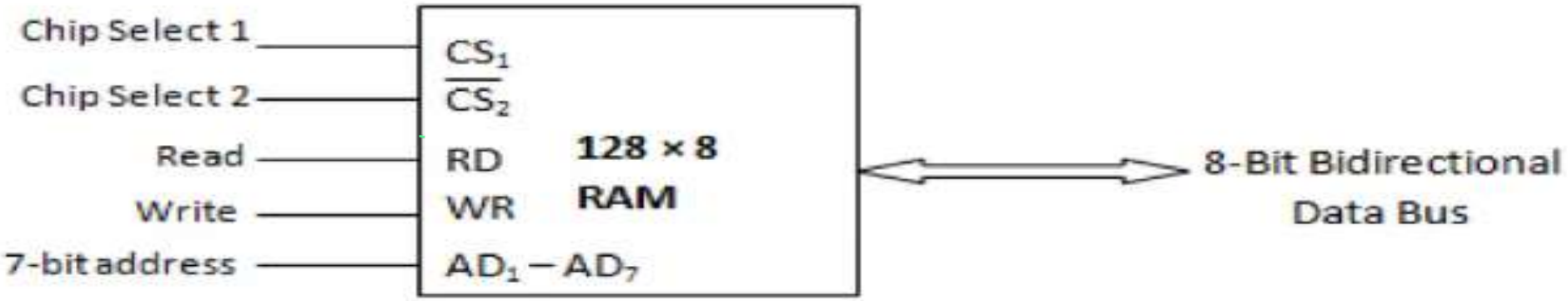
Replace a page that will not be used in near future

FIFO

	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5	2	1	3	4
F3			1	1	1	1	6	6	6	3	3	3	1	1	1	1	1	1	4
F2		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	3	3
F1	3	3	3	3	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5
	X	X	X	✓	X	✓	X	✓	✓	X	✓	✓	X	✓	X	✓	✓	X	X

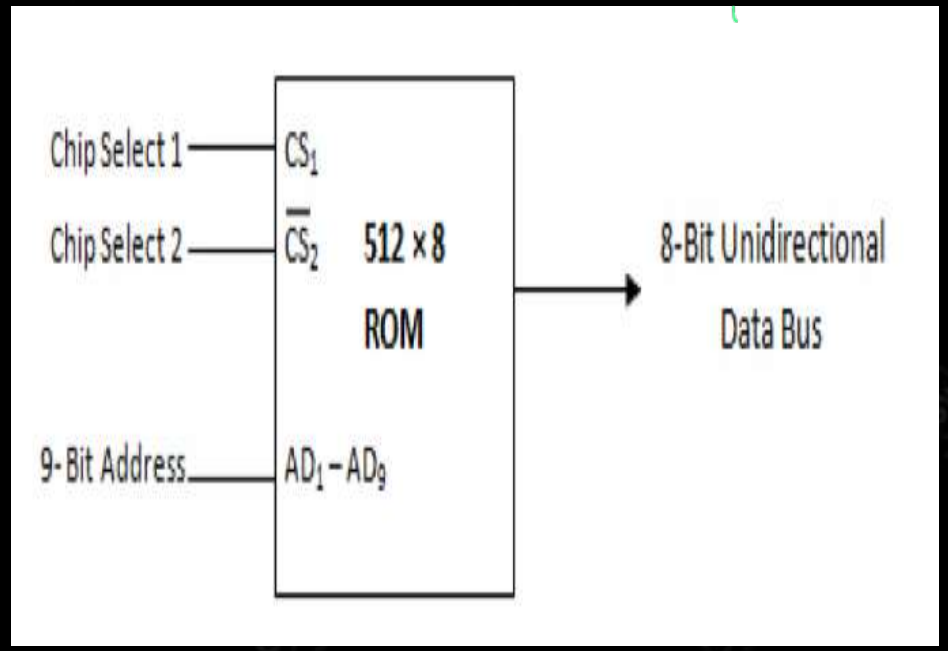
Page fault : 10 Miss Ratio = 10/19 hit Ratio = 9/19

RAM CHIP

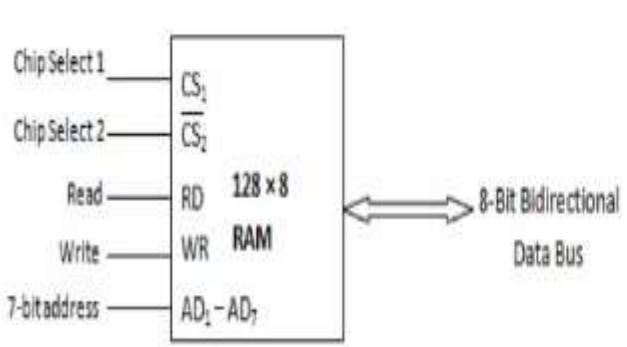


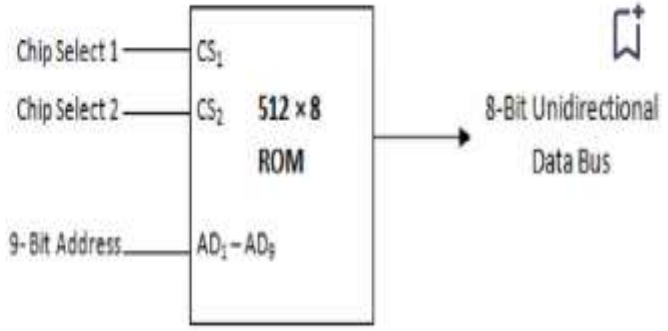
CS_1	CS_2	RD	WR	Memory Operation	State of Data Bus
0	0	X	X	No Operation	High Impedance
0	1	X	X	No Operation	High Impedance
1	0	0	0	No Operation	High Impedance
1	0	0	1	Write Operation	Input Data to RAM
1	0	1	X	Read Operation	Output Data from RAM
1	1	X	X	No Operation	High Impedance

ROM CHIP

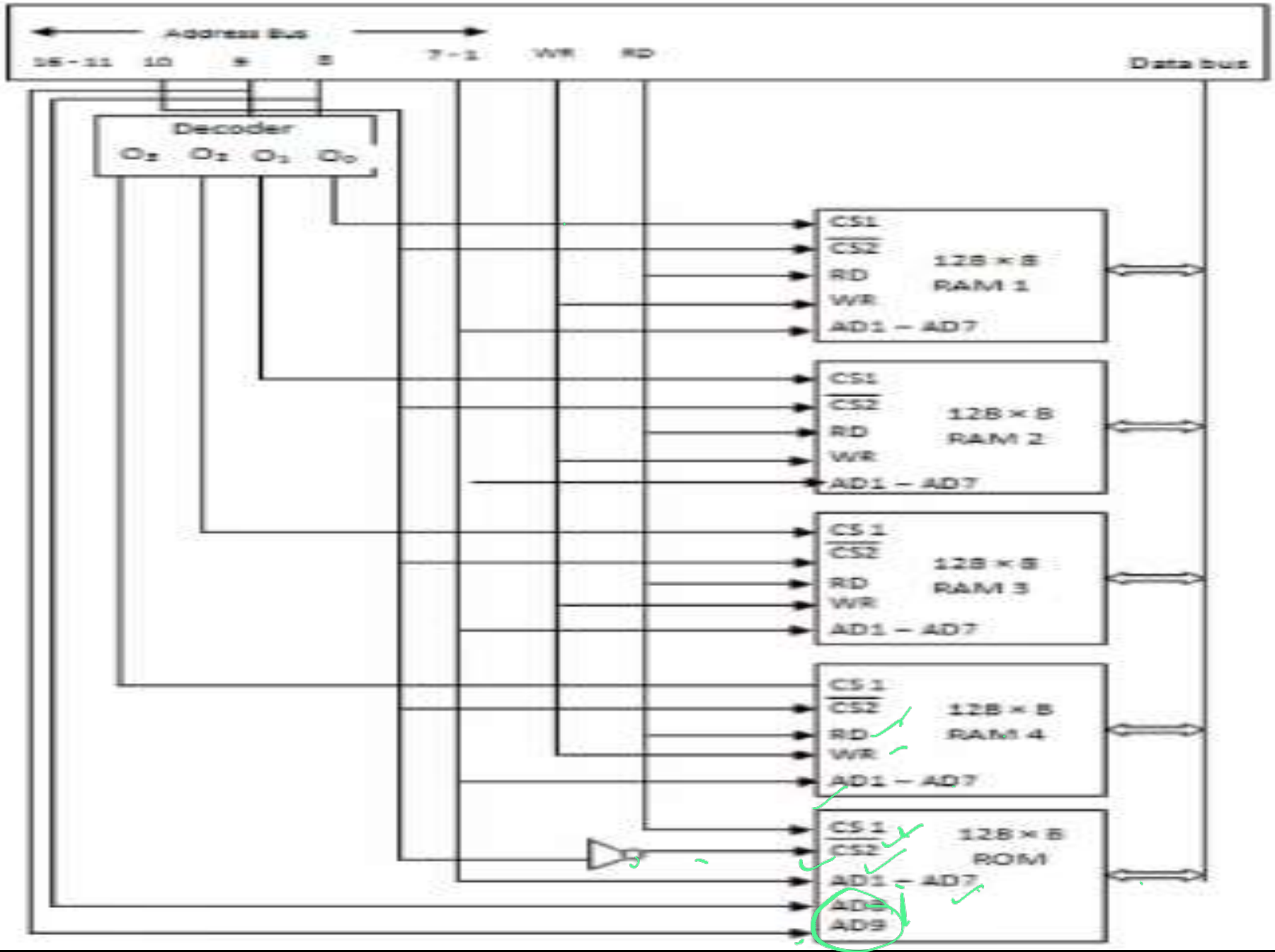


MEMORY ADDRESS MAP





Device Select	Hexadecimal Address	10	9	8	7	6	5	4	3	2	1
RAM 1	0000 – 007F	0	0	0	X	X	X	X	X	X	X
RAM 2	0080 – 00FF	0	0	1	X	X	X	X	X	X	X
RAM 3	0100 – 017F	0	1	0	X	X	X	X	X	X	X
RAM 4	0180 – 01FF	0	1	1	X	X	X	X	X	X	X
ROM	0200 – 03FF	1	X	X	X	X	X	X	X	X	X


Address Bus
15 - 11
10
9
8
7 - 1
WR
RD
Data bus
Decoder
O_3 O_2 O_1 O_0
CS1
CS2
RD
WR
AD1 – AD7
128 × 8
RAM 1
RAM 2
RAM 3
RAM 4
ROM
AD8
AD9

Memory Address Map

512 byte of RAM using 128 byte of RAM chip

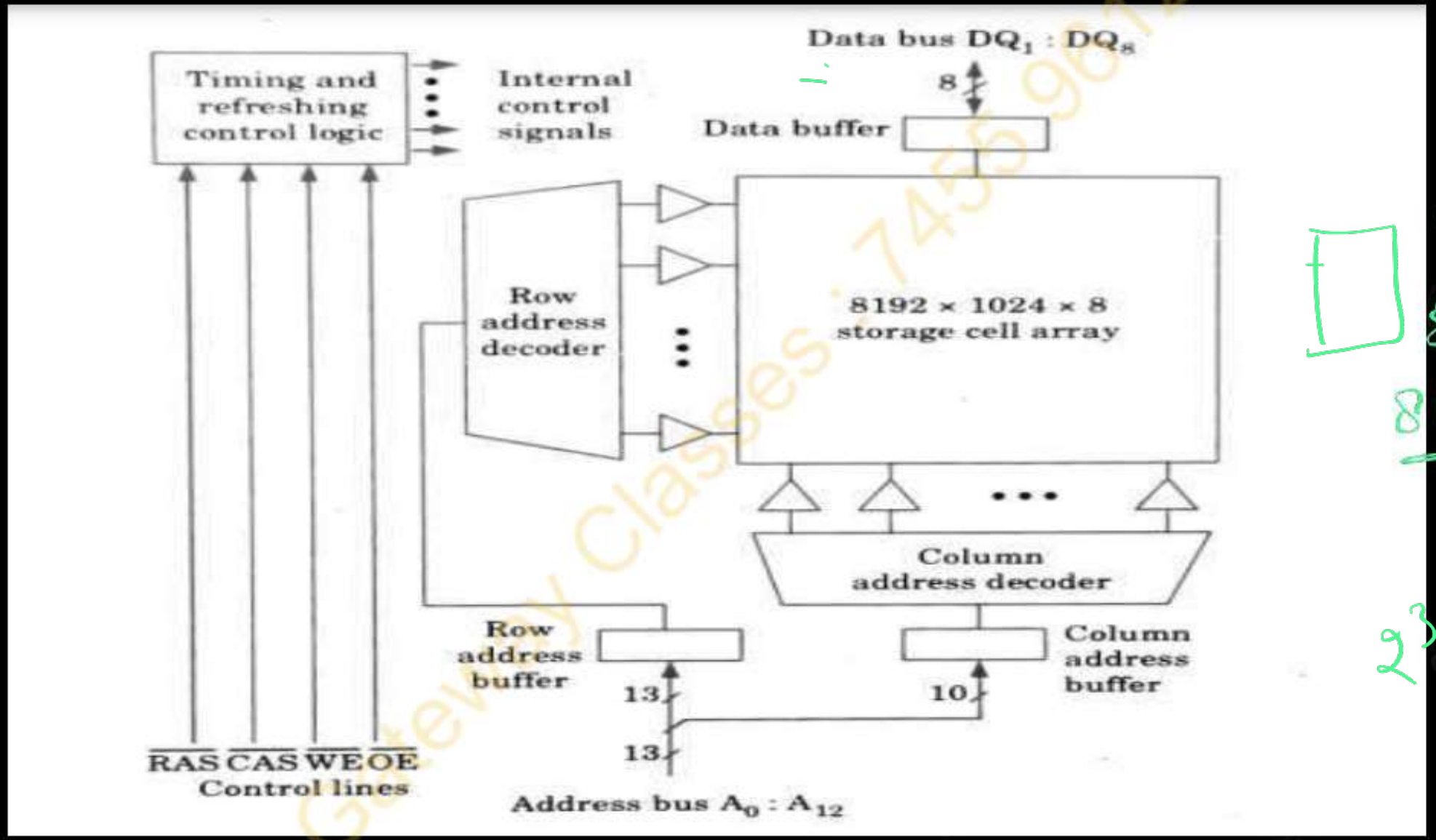
512 byte of ROM using 512 ROM chip

$$\text{Number of RAM chip needed} = \frac{\text{total capacity of RAM}}{\text{Capacity of single chip}}$$

$$\frac{512 \text{ byte}}{128 \text{ byte}} = 4 \text{ chip}$$

$$\text{Number of ROM chip} = \frac{\text{total of capacity of ROM}}{\text{capacity of single chip}} = \frac{512 \text{ byte}}{512 \text{ byte}} = 1 \text{ chip}$$

Give the structure of commercial 8M x 8 bit DRAM chip.



databu

8M x 8

$8 \times 2^{20} \times 8$

$8 \times 2^{10} \times 2^{10} \times 8$

8192 x 1024

↓ Row ↓ colum.

$2^3 * 2^{10}$

2D and 2.5D Memory organization

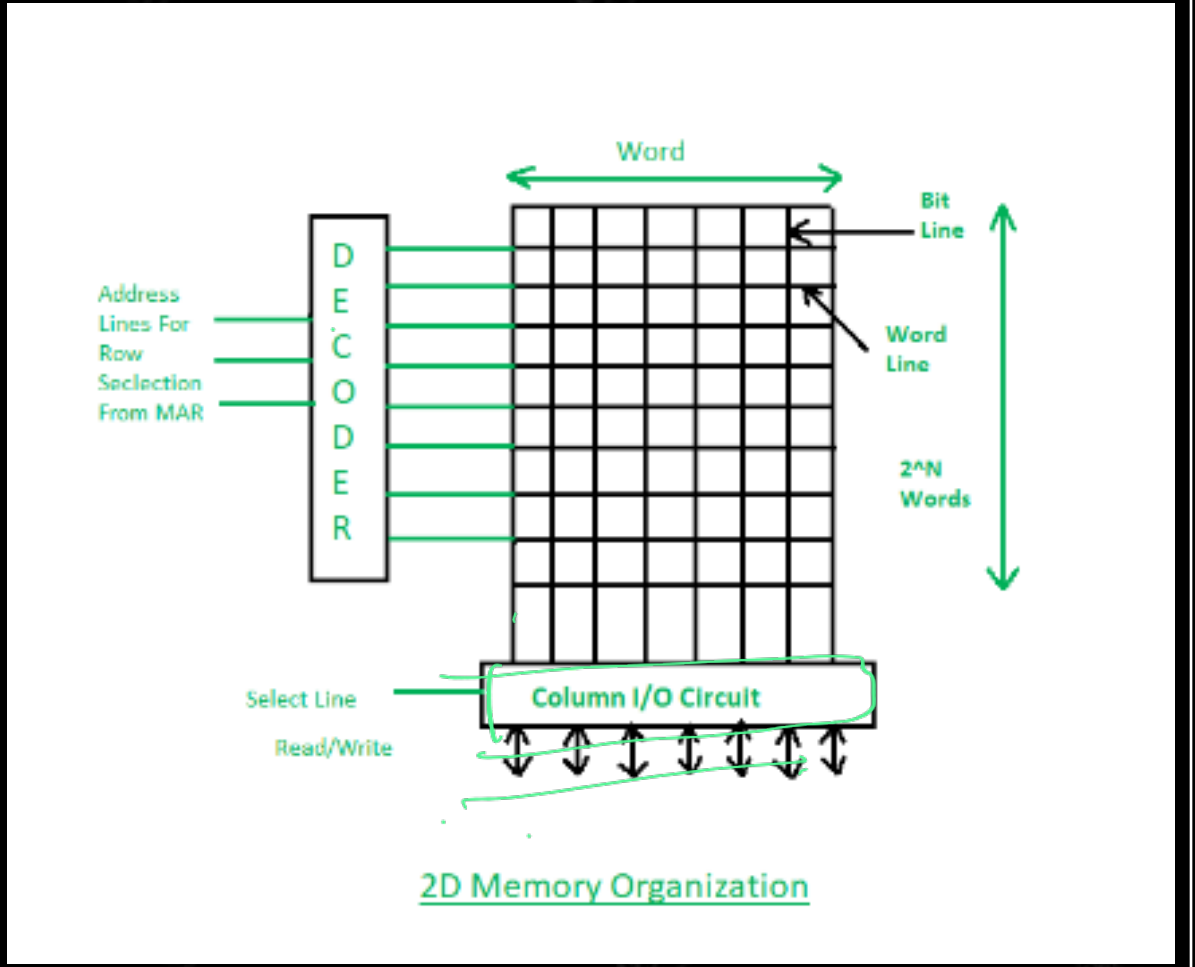
➢The internal structure of Memory either RAM or ROM is made up of memory cells that contain a memory bit. A group of 8 bits makes a byte. The memory is in the form of a multidimensional array of rows and columns. In which, each cell stores a bit and a complete row contains a word. A memory simply can be divided into this below form.

➢2^n = Nwhere n is the no. of address lines and N is the total memory in bytes. There will be 2^n words.

➢**2D Memory organization –**

In 2D organization, memory is divided in the form of rows and columns(Matrix). Each row contains a word, now in this memory organization, there is a decoder. A decoder is a combinational circuit that contains n input lines and 2^n output lines.

➢One of the output lines selects the row by the address contained in the MAR and the word which is represented by that row gets selected and is either read or written through the data lines



2D Memory Organization

2.5D Memory organization –

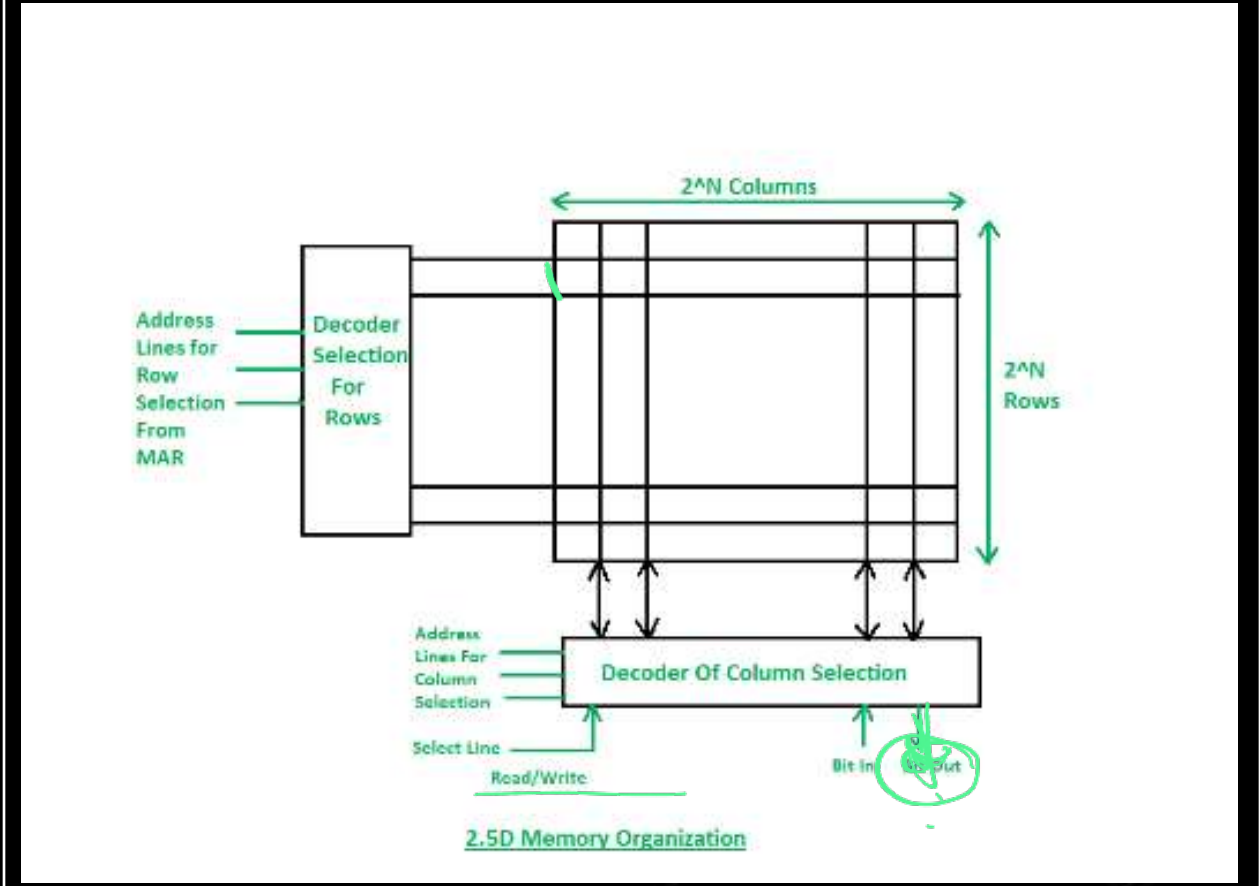
In 2.5D Organization the scenario is the same but we have two different decoders one is a column decoder and another is a row decoder. Column decoder is used to select the column and a row decoder is used to select the row. The address from the MAR goes as the decoders' input. Decoders will select the respective cell through the bit outline, then the data from that location will be read or through the bit, inline data will be written at that memory location.

Read and Write Operations –

➢If the select line is in Reading mode then the Word/bit which is represented by the MAR will be available to the data lines and will get read.

➢If the select line is in write mode then the data from the memory data register (MDR) will be sent to the respective ➢cell which is addressed by the memory address register (MAR).

➢With the help of the select line, we can select the desired data and we can perform read and write operations on it.



2.5D Memory Organization

Associative memory or Content Addressable Memory (CAM).

- An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.

- When a write operation is performed on associative memory, no address or memory location is given to the word. The memory itself is capable of finding an empty unused location to store the word.
- , when the word is to be read from an associative memory, the content of the word, or part of the word, is specified. The words which match the specified. The words which match the specified content are located by the memory and are marked for reading

- Associative memory is access simultaneously and in parallel on the basis of data content rather by specific address location
- Because of its organization the associative memory is uniquely entitled to do parallel search by data association
- An associative memory is more expensive than RAM because each cell have storage capacity as well as logic circuit for matching its content with an external arguments
- An associative memory consists of a memory array and logic for 'm' words with 'n' bits per word.
- The functional registers like the argument register A and key register K each have n bits, one for each bit of a word. The match register M consists of m bits, one for each memory word

ARGUMRENT REGISTER(A)

It contains the words to be searched

It has n bits (one for each bit of word)

KEY REGISTER(K)

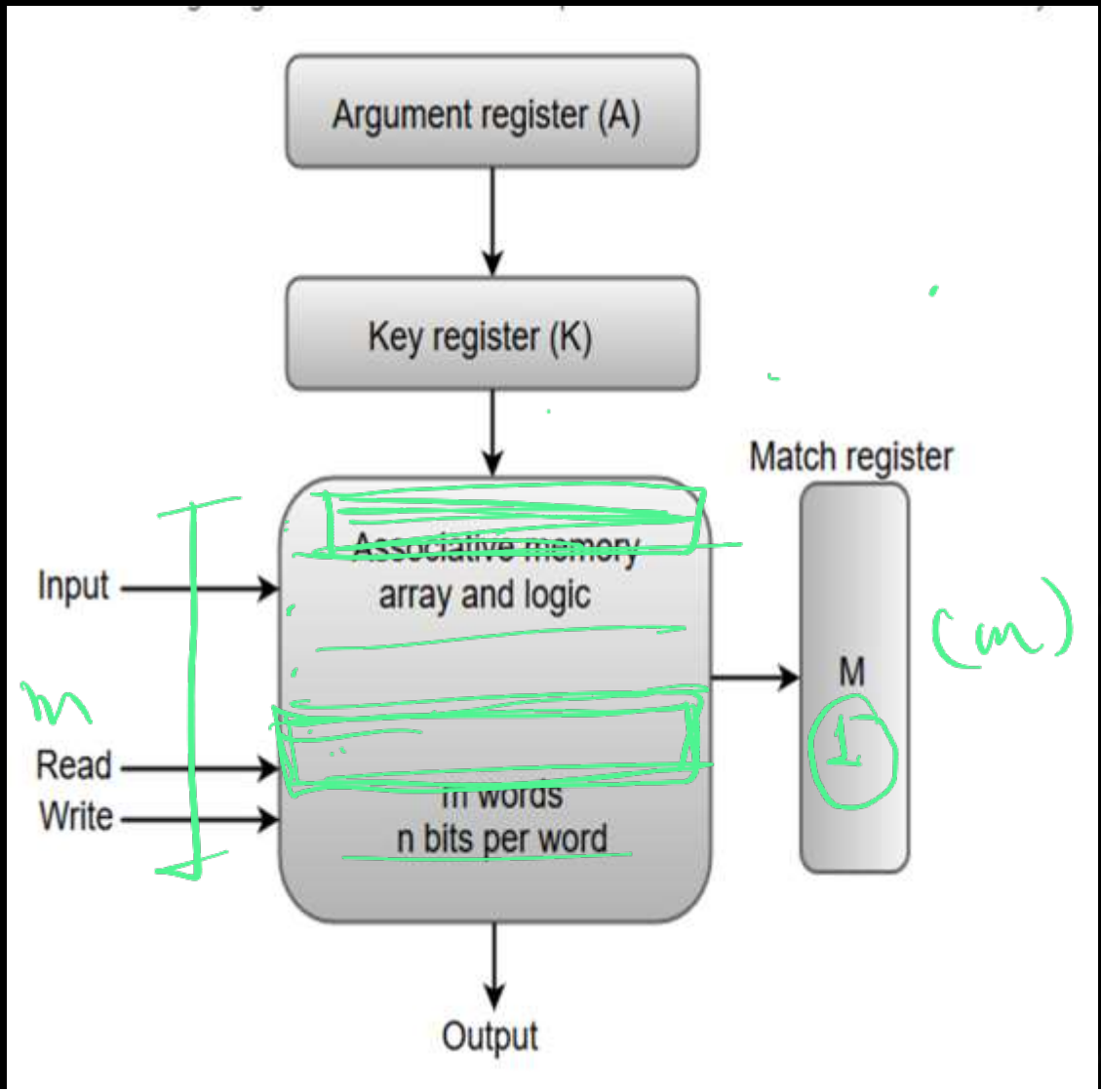
The key register (K) provides a mask for choosing a particular field or key in the argument word.

If the key register contains a binary value of all 1's, then the entire argument is compared with each memory word. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared. Thus, the key provides a mask for identifying a piece of information which specifies how the reference to memory is made.

Associative memory array and logic

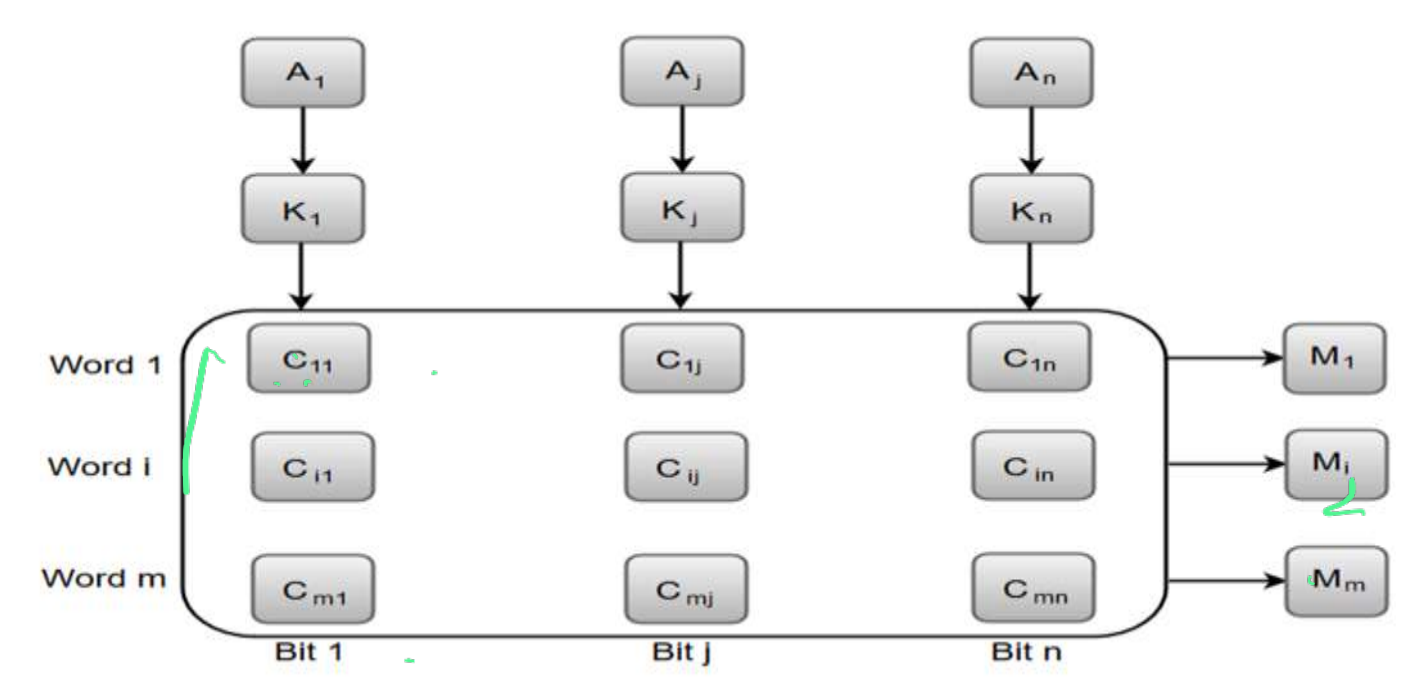
It contain the word that are to be compared with the

- the argument word in parallel
- It consists of m words with n bits per words



Match logic(M)

- It has m bits, one bits corresponding to each word in the memory array.
- After the matching process, the bits corresponding to matching words in match register are set to 1 .Searching done in parallel manner
- Reading is accomplished by the sequential access in memory for these words whose match bits set one



AUXILLARY MEMORY

- An Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. It is where programs and data are kept for long-term storage or when not in immediate use.

1.Magnetic Disks

- A magnetic disk is a type of memory constructed using a circular plate of metal or plastic coated with magnetized materials. Usually, both sides of the disks are used to carry out read/write operations. However, several disks may be stacked on one spindle with read/write head available on each surface.

MAGNETIC TAPE

- Magnetic tape is a storage medium that allows data archiving, collection, and backup for different kinds of data. The magnetic tape is constructed using a plastic strip coated with a magnetic recording medium.
- The bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.

FLASH MEMORY

- Flash memory is a long-life and non-volatile storage chip that is widely used in embedded systems. It can keep stored data and information even when the power is off. It can be electrically erased and reprogrammed.
- Flash memory was developed from EEPROM (electronically erasable programmable read-only memory

Memory interleaving(memory module)

- A single memory module cause sequential access, only one memory access is possible a time so throughput is low
- Memory interleaving is the technique to increase the throughput Here system is divided into a number of independent modules which answer read and write request independently in parallel.
- Interleaved memory is designed to compensate for the relatively slow speed of dynamic random-access memory (DRAM) or core memory by spreading memory addresses evenly across memory banks. In this way, contiguous memory reads and writes use each memory bank, resulting in higher memory throughput due to reduced waiting for memory banks to become ready for the operation

two types of memory interleaving

Higher order interleaving

- Consecutive address are stored within in a same module
- It uses higher order bits(MSB) as the module address and lower order bits as word within each module

	Module 00
00	10
01	20
10	30
11	40

	Module 01
00	50
01	60
10	70
11	80

	Module 10
00	90
01	100
10	110
11	120

	Module 11
00	130
01	140
10	150
11	160

Lower order bits / Higher order bits	
0000	10
0001	20
0010	30
0011	40
0100	50
0101	60
0110	70
0111	80
1000	90
1001	100
1010	110
1011	120
1100	130
1101	140
1110	150
1111	160

ADVANTAGES

- easy memory extension by addition of one or more memory module
- Provides better reliability since failed module affect only a localized area of the address space.

DISADAVANTAGES

- When a consecutive location are accessed only one module is involved other sir idle
- Scheme will cause conflicts in case of pipelined vector processing due to sequentially of instruction
- This sequence is useful in single user system
- **Lower order interleaving**
- Consecutive address stored in a consecutive module
- It uses low order bit as a module address and higher order bit as word address within each module

ADVANTAGES

- Faster access to a block of data and higher utilization of the memory system as whole

Disadvantage

- A failure of single module cause whole system fail

	Module 00
00	10
01	50
10	90
11	130

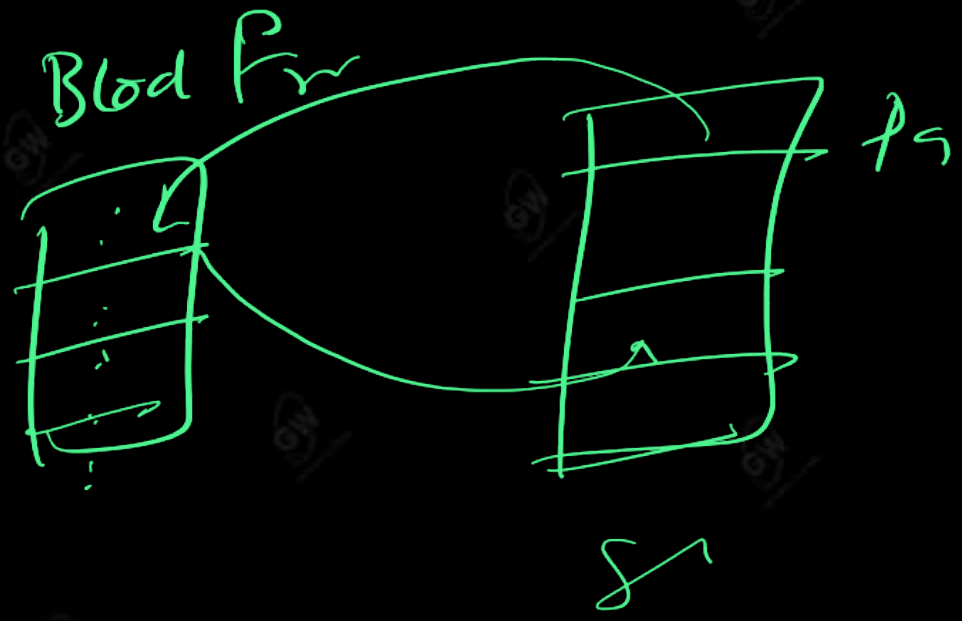
	Module 01
00	20
01	60
10	100
11	140

	Module 10
00	30
01	70
10	110
11	150

	Module 11
00	40
01	80
10	120
11	160

PAGING

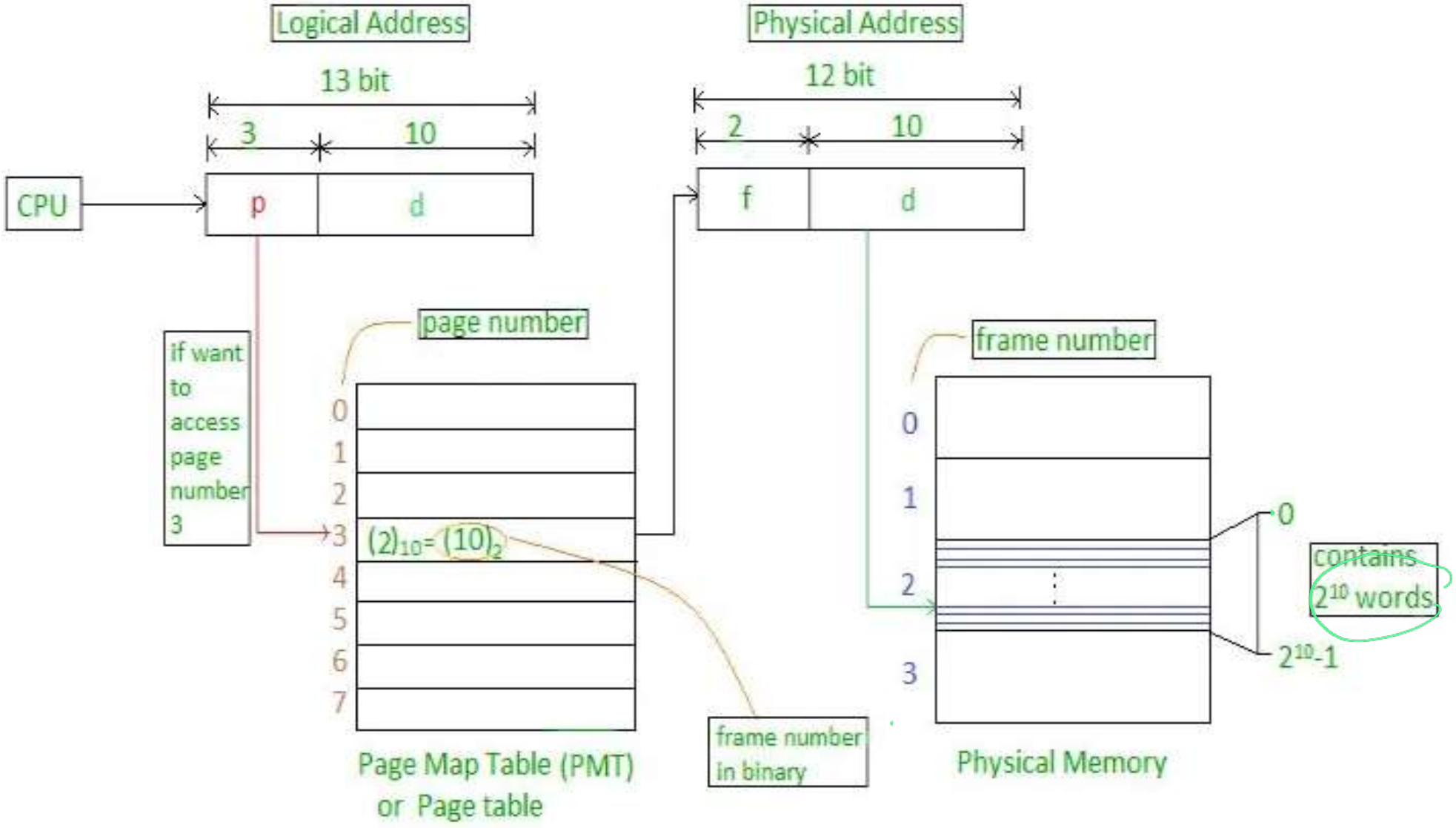
- Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. Additionally, frames will be used to split the main memory. This scheme permits the physical address space of a process to be non – contiguous
- In paging, the physical memory is divided into fixed-size blocks called page frames, which are the same size as the pages used by the process. The process's logical address space is also divided into fixed-size blocks called pages, which are the same size as the page frames. When a process requests memory, the operating system allocates one or more page frames to the process and maps the process's logical pages to the physical page frames
- The mapping between logical pages and physical page frames is maintained by the page table, which is used by the memory management unit to translate logical addresses into physical addresses. The page table maps each logical page number to a physical page frame number.



- The mapping from virtual to physical address is done by the Memory Management Unit (MMU) which is a hardware device and this mapping is known as the paging technique.
- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called frames.
- The Logical Address Space is also split into fixed-size blocks, called pages.
- Page Size = Frame Size
- Let us consider an example:
- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)

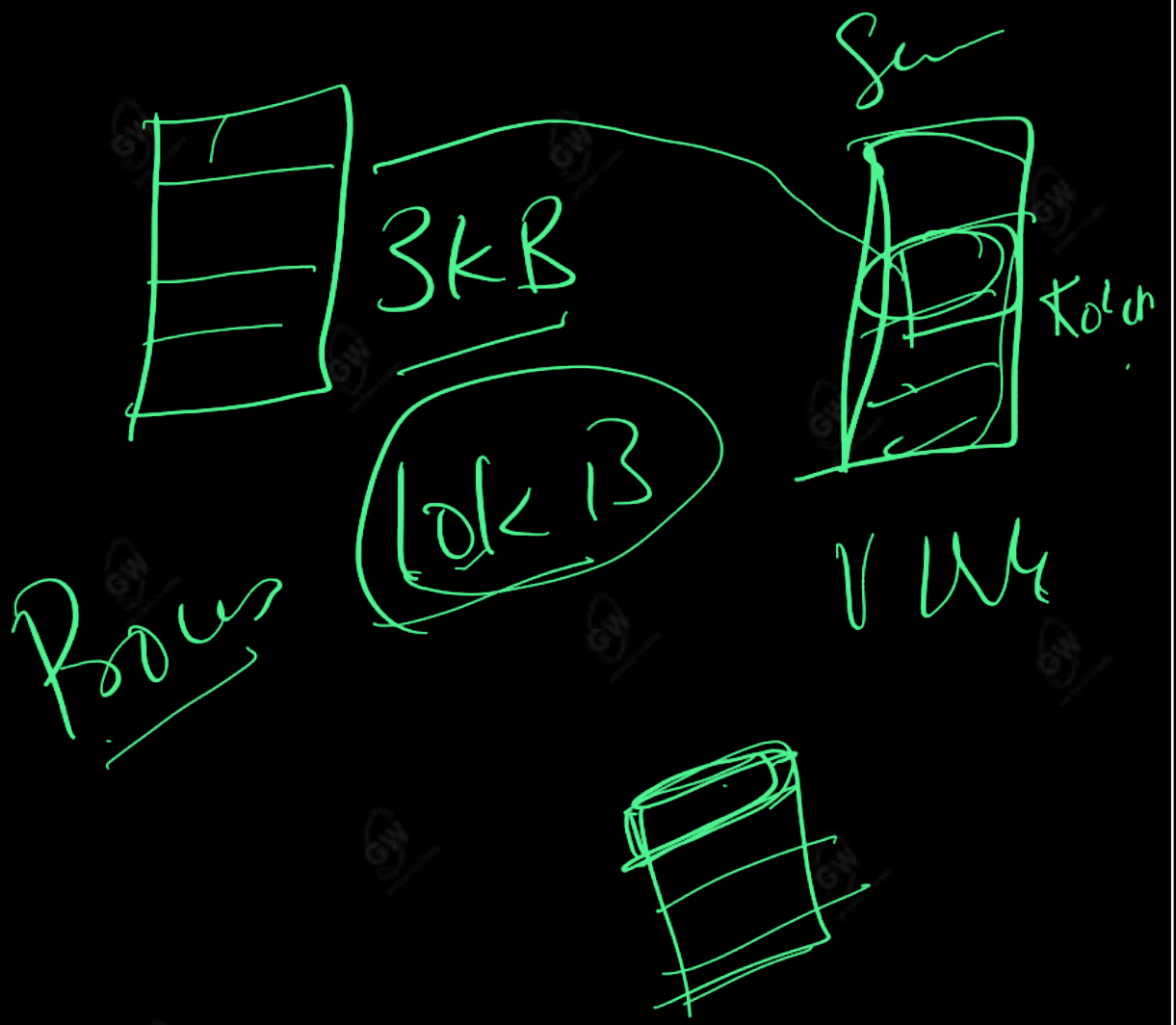
Number of frames = Physical Address Space / Frame size = 4 K / 1 K = 4 = 2^2

Number of pages = Logical Address Space / Page size = 8 K / 1 K = 8 = 2^3



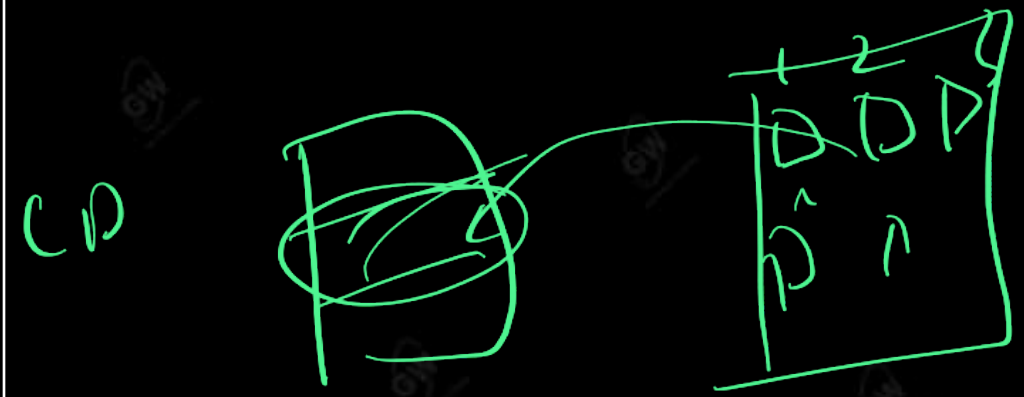
VIRTUAL MEMORY

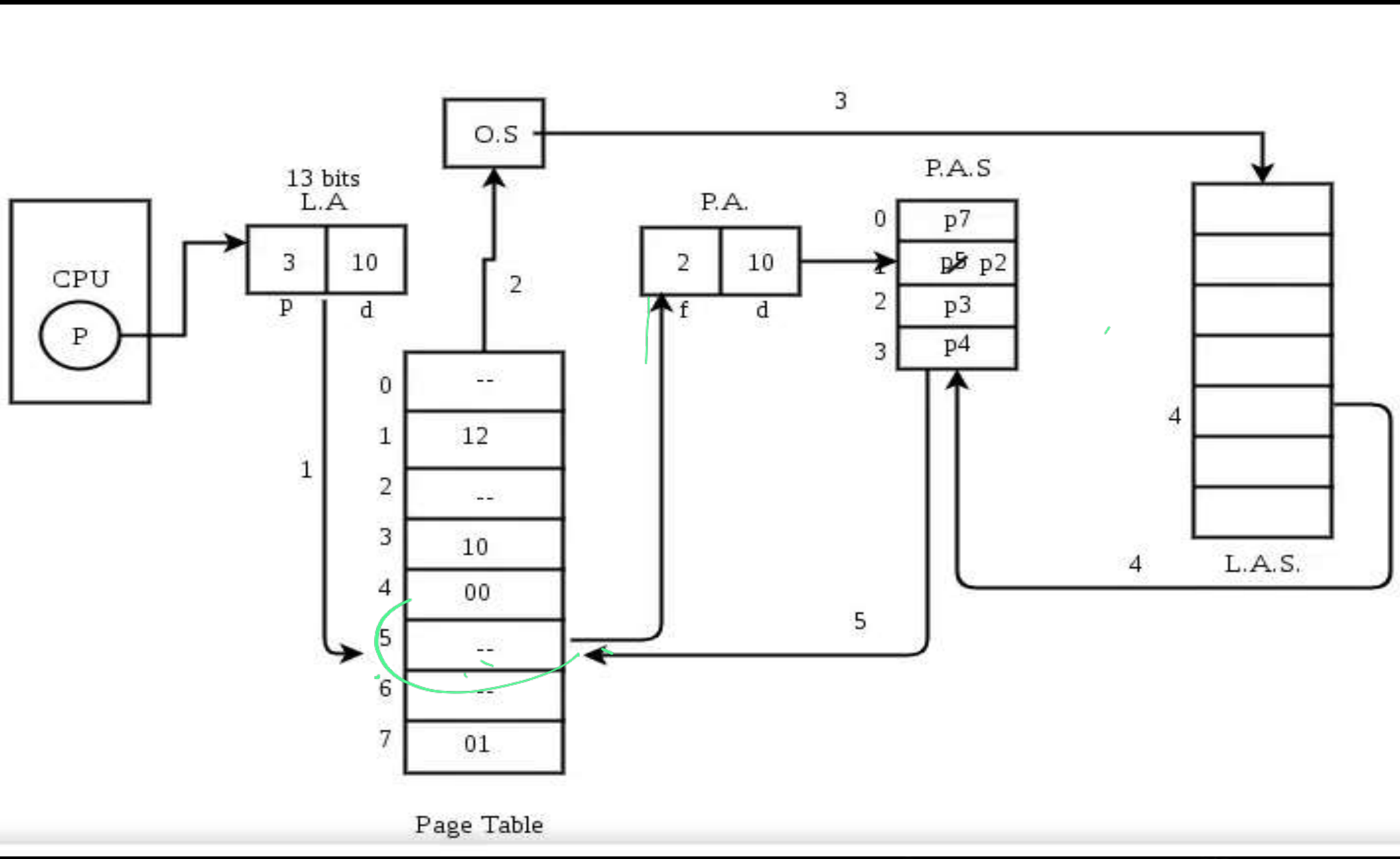
- Virtual memory is a memory management technique used by operating systems to expand the available memory of a computer beyond its physical RAM. It allows a computer to compensate for shortages of physical memory by temporarily transferring data from RAM to disk storage. When the data is needed again, it is transferred back from the disk to RAM. This process is transparent to the user and allows programs to run as if they have more memory than is actually available.



Demand Paging

- . The process of loading the page into memory on demand (whenever a page fault occurs) is known as demand paging. The process includes the following steps are as follows:
- If the CPU tries to refer to a page that is currently not available in the main memory, it generates an interrupt indicating a memory access fault.
- The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.
- The OS will search for the required page in the logical address space.
- The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision-making of replacing the page in physical address space.
- The page table will be updated accordingly.
- The signal will be sent to the CPU to continue the program execution and it will place the process back into the ready state.




O.S
3
13 bits
L.A
CPU
P
3
10
p
d
2
P.A.
2
10
f
d
P.A.S
0
p7
1
p5
p2
2
p3
3
p4
4
L.A.S.
0
--
1
12
2
--
3
10
4
00
5
--
6
--
7
01
1
2
3
4
5
Page Table