Unit :- 04

Memory

Page Replacement Algorithm

FIFO

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
				H							H	H			H	H			

Page hit = 5

Page Fault = 20 - 5
= 15

LRU

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
				H		H				H	H		H		H		H	H	

Page hit = 8

Page fault = 20 - 8
= 12

Optimal

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
				H		H		H	H		H	H		H	H	H		H	H

Page Hit = 11 fault = 20 - 11 = 9

(2.)

	1	2	3	4	5	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2
FIFO	1	1	1	1	5	5	5	5	5	5	5	8	8	8	8	8	8	8	8	8	8	8	2
		2	2	2	2	2	2	2	1	1	1	1	1	1	9	9	9	9	9	9	9	9	9
			3	3	3	3	3	3	3	6	6	6	6	6	6	6	6	5	5	5	5	5	5
				4	4	4	4	4	4	4	7	7	7	7	7	7	7	7	4	4	4	4	
						(H)	(H)	(H)					(H)	(H)		(H)	(H)	(H)			(H)	(H)		

Page hit = 10

Page fault = 23 - 10
= 13

	1	2	3	4	5	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2
LRU	1	1	1	1	5	5	5	5	5	6	6	6	6	6	6	6	6	6	5	5	5	5	5
		2	2	2	2	2	2	2	1	1	1	1	1	1	9	9	9	9	9	9	9	9	9
			3	3	3	3	3	3	3	3	7	7	7	7	7	7	7	7	7	4	4	4	4
				4	4	4	4	4	4	4	4	8	8	8	8	8	8	8	8	8	8	8	2
						(H)	(H)	(H)					(H)		(H)	(H)					(H)	(H)	

Page hit = 8

Page fault = 23 - 8
= 15

	1	2	3	4	5	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2
Optimal	1	1	1	1	1	1	1	1	1	1	7	7	7	7	7	7	7	7	7	4	4	4	4
		2	2	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
			3	3	3	3	3	3	3	6	6	8	8	8	8	8	8	8	8	8	8	8	2
				4	4	4	4	4	4	4	4	4	4	9	9	9	9	9	9	9	9	9	9
						(H)	(H)	(H)	(H)				(H)	(H)		(H)	(H)	(H)	(H)			(H)	(H)	

Page Hit = 12

Page fault = 23 - 12
= 11

③

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FIFO	1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
		2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
			3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3
					(H)						(H)				(H)			(H)		

no. of hits = 4

page faults = 20 − 4

= 16

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
LRU	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
	1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
		2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
			3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6
					(H)						(H)				(H)			(H)	(H)	

no. of hits = 5

page faults = 20 − 5

= 5

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
optimal	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3
		2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
			3	4	4	4	5	6	6	6	6	6	6	6	6	7	1	1	1	6
					(H)	(H)			(H)	(H)	(H)				(H)	(H)		(H)	(H)	

no. of hits = 9

page faults = 20 − 9

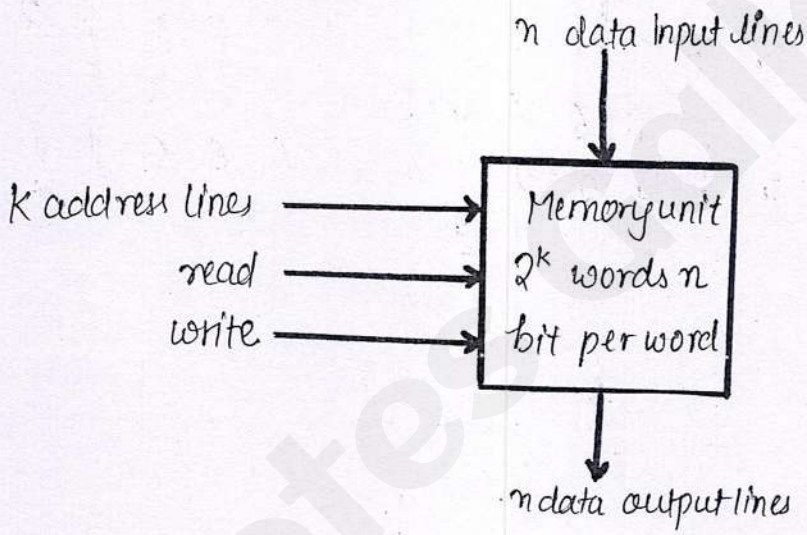
= 11

Memory:- It is an electronic circuit that allow data to be stored and retriere when required.

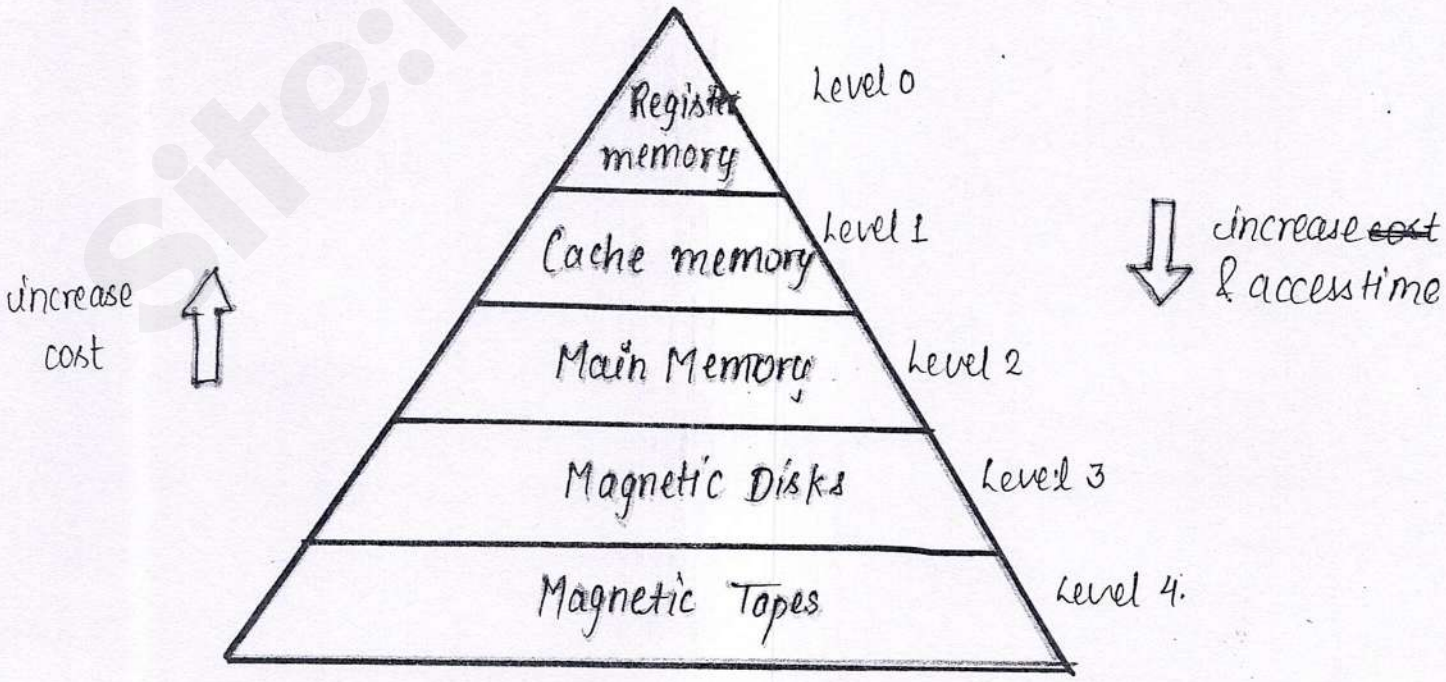
Memory unit that communicate directly with the CPU is known as main memory.

The storage device that provides backup storage is known as auxiliary memory.

Memories are made up with register, each register holds (store 1-bit data. Each location in memory is identified by its address. The total no. of bits that a memory can store is its capacity.



Memory Hierarchy



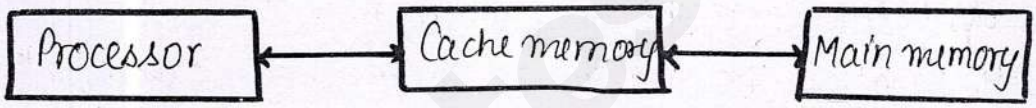
As moving down in the hierarchy, following occurs:-

1. Increasing capacity
2. Increasing access time
3. Decreasing cost per bit

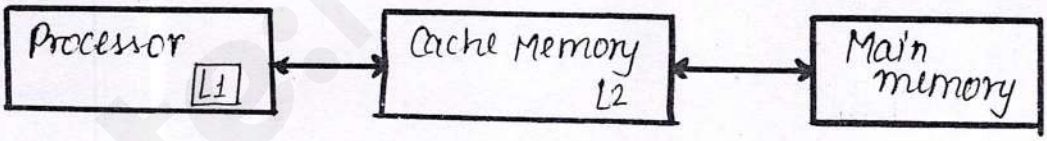
Thus, we can say that memory devices at lower level is faster to access smaller in size as compared to higher level.

Cache Memory:- It is defined as a very high speed memory used in computer system to compensate, the speed difference blw the main memory access time.

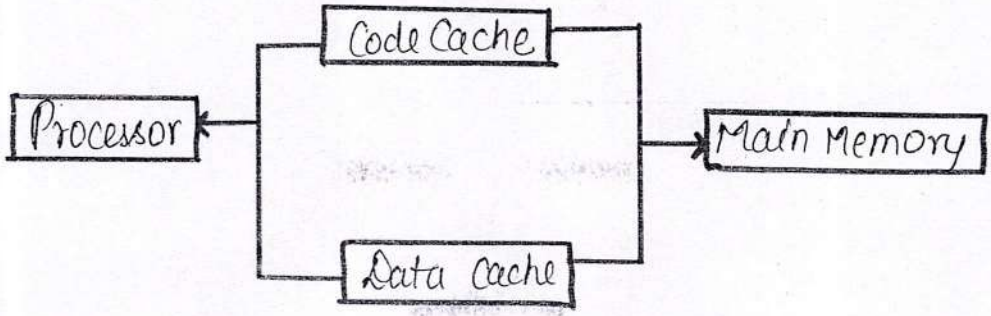
A very high speed memory is cache memory is used to increase the speed of processing by making current program and data available to CPU at rapid rate.



It is also possible to place a smaller cache blw cache memory and processor. This new cache resides in processor.



L1 is faster cache because it resides in a processor and smaller in size. Another cache system is the split cache that required two cache memories. In this cache, a processor will use one cache to store code and another one store data.



Split Cache Organization

When the CPU needs to access memory, the cache is examined first if the word is found in cache is known as <u>cache hit</u>, if it is not available in cache then the desired memory block is copied in cache memory then it is used. this condition referred as <u>cache Miss</u>.

The percentage of accesses where the processor finds the code or data word it needs, in the cache memory is called Hit Rate or Hit Ratio.

$$\text{Hit Rate} = \frac{\text{No. of hit}}{\text{Total No. of Cycle}} * 100$$

Ques. The application program in a computer system with cache uses 400 instructions acquisition cycle from cache memory and 100 from main memory. what is the hit rate? If the cache memory operates with zero wait state and the main memory bus cycle use three wait state what is the avg. no. of wait states during the program execution?

Sol":-

no. of hits = 400

total no. of cycle = 400 + 100

$$\text{Hit rate} = \frac{400}{400+100} * 100$$

$$= \frac{400}{500} \times 100$$

Hit Rate = 80%.

Total no. of wait state = 400 * 0 + 100 * 3 = 300

$$\text{Avg. wait state} = \frac{\text{Total no. of wait}}{\text{Total no. of cycle}}$$

$$= \frac{300}{500}$$

$$= 0.6$$

There are various design issues in cache memory such as cache size, mapping techniques, block size, replacement algorithm and no. of words per block.

Mapping Techniques in Cache

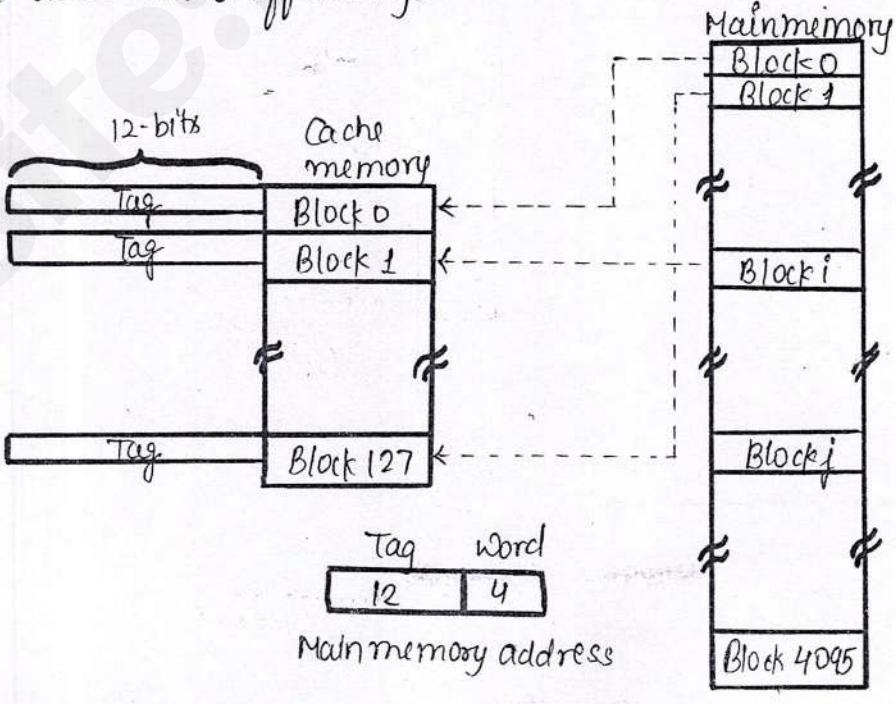
Mapping process is a method of transferring information from main memory to cache memory. There are three mapping process :-

1. Associative mapping process / Content-Based mapping
2. Direct mapping
3. Set Associative mapping

Associative mapping (Fully Associative mapping)/ Content-Based

In this technique, a main memory block can be placed into any cache block position as there is no fix block, the memory address has only two fields :- word and tag.

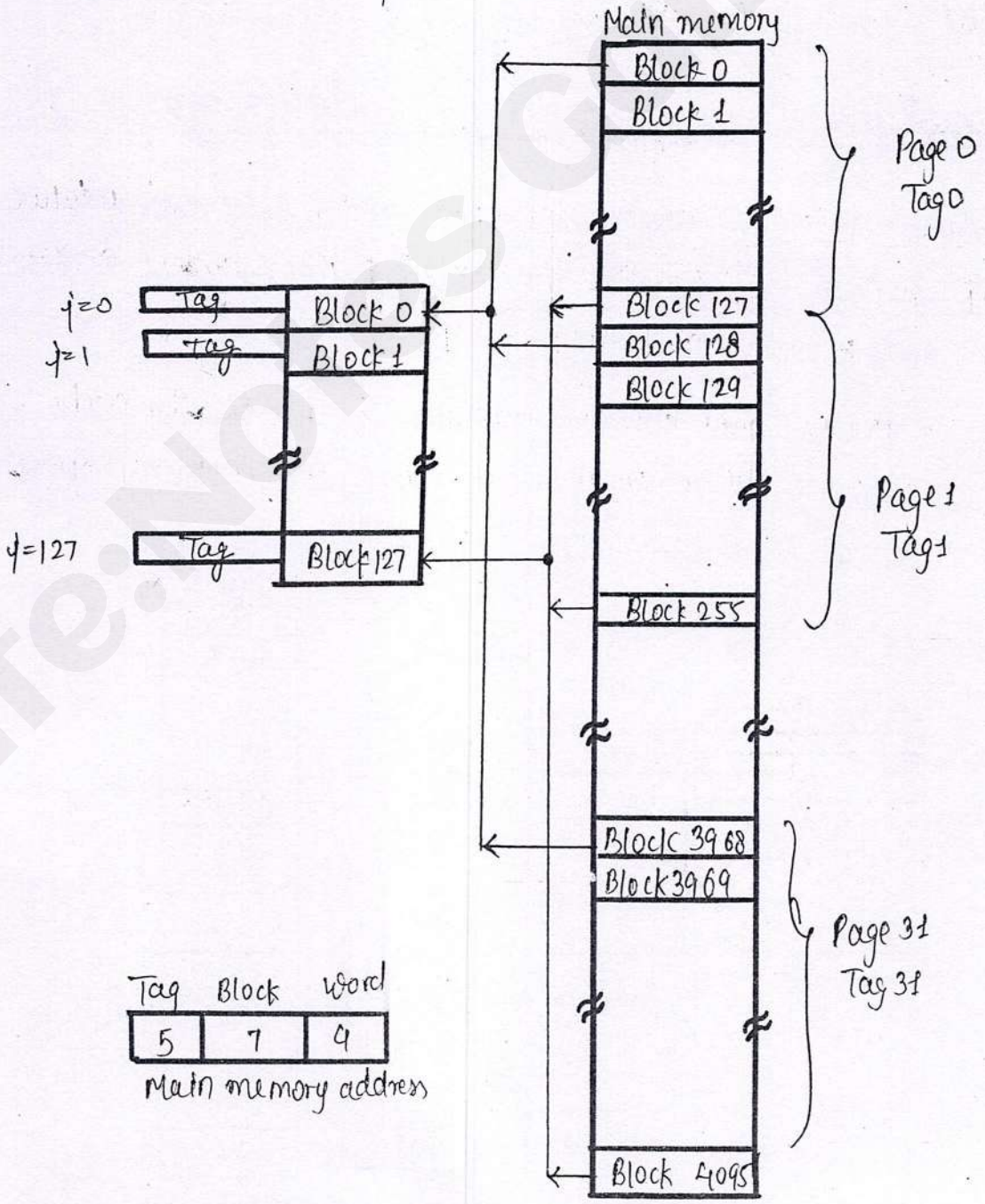
This technique gives complete freedom to choose the cache location in which to place the memory block. Thus, the memory space in the cache can be used more efficiently.



Associative-mapped cache

Direct Mapping:- It is the simplest mapping technique. In this technique, each block from the main memory has only one possible location in the cache organization. To implement such cache system, the address is divided into three fields - word field, block field and tag field.

The main drawback of direct map cache is that if processor needs to access same memory locations from two different pages of the main memory frequently, since only one of these locations can be in the cache at a time. Therefore, we can say that direct map cache is easy to implement but it is not flexible.



S.No	Feature	Direct Mapping	Associative Mapping	Set-Associative
1.	Mapping function	Simple modulo operation. (index)	Content-addressable memory (CAM).	Combination of direct and associative mapping, using a set index and associativity within the set.
2.	No. of entries per set	1 (single)	Number of cache lines.	More than 1 (typically a power of 2).
3.	Flexibility	Limited flexibility due to a fixed mapping.	Flexible, any block can be placed anywhere in the cache.	Combine some flexibility with a structure for efficient searching.
4.	Search Mechanism	Direct access to a specific cache line using index.	Full search across all cache lines simultaneously.	Search limited to a specific set, then direct access within the set.
5.	Cache Hit Time	Fast, due to direct mapping	Slower than direct mapping, but faster than set associative.	Moderate, falls b/w direct and fully associative mapping
6.	Cache Miss Handling	Simple, no competition for cache lines.	complex due to competition for cache lines among all blocks	Moderate complexity. limited competition within a set.
7.	Cost and complexity	Low cost, simple hardware	Higher cost, more complex hardware	Moderate cost and complexity.
8.	Example Use Case.	Small embedded systems or limited-resource environments.	General-purpose systems with larger caches.	Commonly used in modern processors for a balance b/w flexibility and performance.

Difference b/w DRAM and SRAM

DRAM	SRAM
Constructed of tiny capacitors that leak electricity.	Constructed of circuits similar to D flip-flops.
Requires a recharge every few milli-seconds to maintain its data.	Holds its contents as long as power is available.
Inexpensive	Expensive.
Slower than SRAM.	Faster than DRAM.
Can store many bits per chip.	Cannot store many bits per chip.
Uses less power.	Uses more power.
Generates less heat	Generates more heat.
Used for main memory	Used for cache.
Simple Structure - has a transistor and a capacitor.	Complex structure - has flip flops.
Has a higher density.	Has a lower density.

Set Associative Mapping: The set associative mapping is a combination of direct and associative mapping.

It contains several groups of direct map blocks that operate as several direct map cache in parallel.

The disadvantage of direct mapping is the two word of the same index but different tag cannot be stored at the same time into ~~index but~~ cache. As an improvement to this disadvantage, a third type of

cache organization called set associative mapping.
It is most flexible and give high hit ratio.
It is costly to implement and here tag length is increased

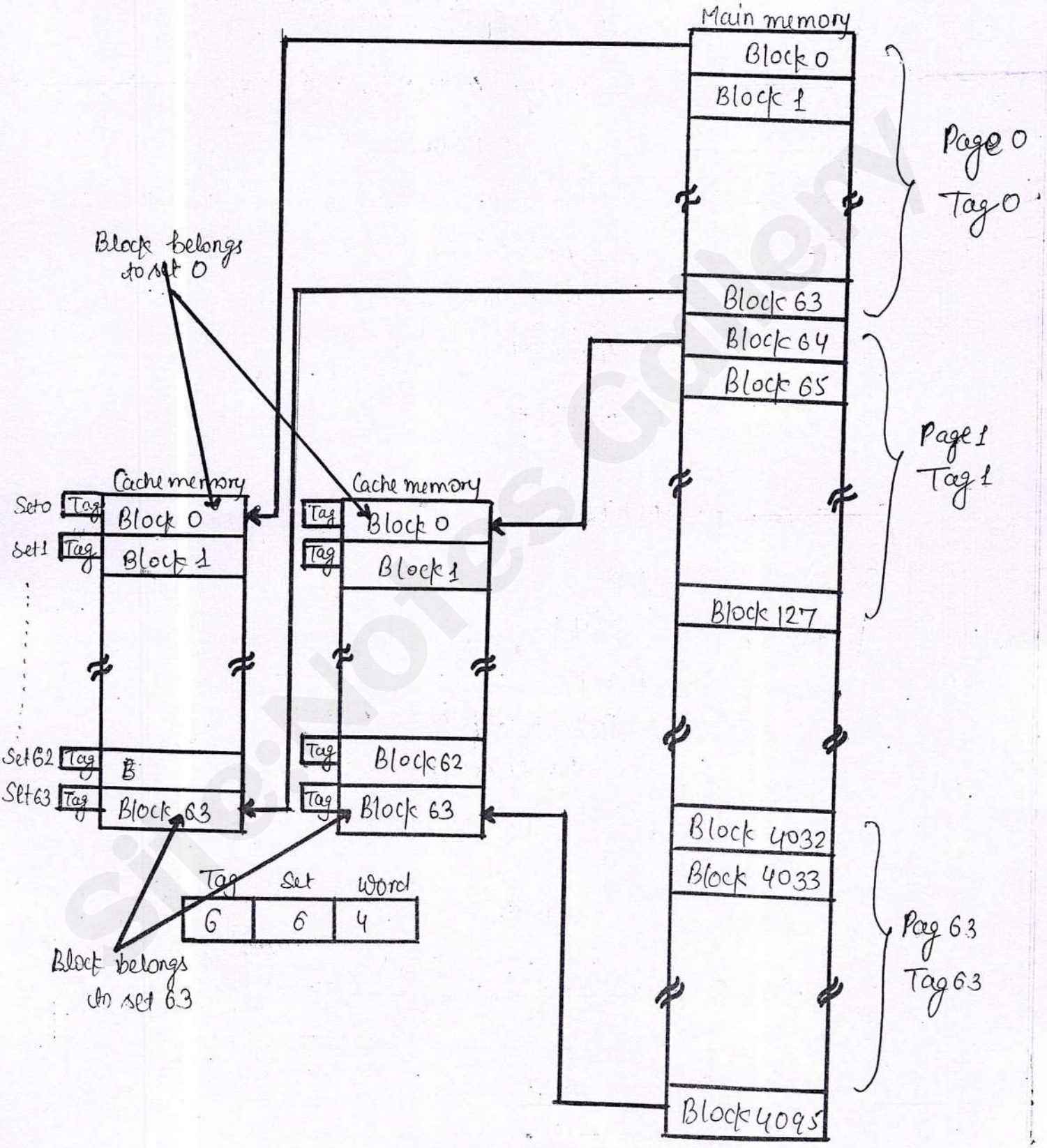


Fig:- Two-way set Associative Cache

Ques. A block set associative cache consist of 64 blocks divided into 4 block set. The main memory contains 4096 blocks, each consist of 128 words of 16 bit length.

1. How many bits are there in main memory.
2. How many bits are there in each of the tag, set and word field.

Sol^n ① no of bits in main memory = $4096 * 128 * 16$

$= 8388608$ bits

② main memory size = No. of Block * No. of words per block

$= 4096 * 128$

$= 4K * 2^7$

$= 2^{12} * 2^7$

$= 2^{19}$

MM bit = 19 bit

No. of words = 128

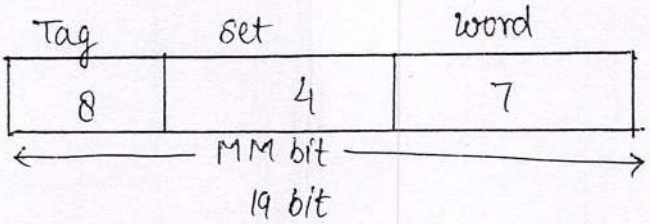
word bit = 2^7 = 7 bit

cache size = 64 block

$$\text{No. of set} = \frac{\text{Total no. of block in cache}}{\text{No. of sets per block}}$$

$$= \frac{64}{4} = 16$$

set bit = 2^4 = 4 bit



Tag = MM bit − (w bit + set bit)

= 19 − (4+7)

Tag = 8

Ques A digital computer has a memory unit of 64k * 16 and a cache memory of 1k words. The cache uses direct mapping with a block size of 4-words. How many bits are there in tag index, block and word field of the address format?

Sol"→ Main memory size = 64 K * 16

$= 64 \times 2^{10} * 16$

$= 2^6 \times 2^{10} * 16$

MM bit $= 2^{16} * 16$

MM bit = 16 bits

Cache memory size = 1k = 1024.

no. of word bits = no. of words

$= 4$

$= 2^2$

word bit = 2

$$\text{no. of block bits} = \frac{\text{cache size}}{\text{no. of words per block}}$$

$= \frac{1024}{4}$

$= 256$

$= 2^8$

no. of block bits = 8 bits

No. of Tag bit = 16 − (2+8)

= 16 − 10

Tag bit = 6

Tag	Block	word
6	8	2

16-bit

Que Ques:- A two (no. of set per block) way set associative cache uses block of 4 words. The cache can accomodate a total of 2048 words from the main memory. The main memory size is 128K*32. How many bits are there in tag, set and word in address format?

Sol^n:- Main memory size = 128K * 32

$= 2^7 * 2^{10} * 32$

$= 2^{17} * 32$

MM bit = 17 bits

no. of word bits = no. of words

$= 4 \Rightarrow 2^2$

word bit = 2 bit

set bit =

$$\text{no. of set} = \frac{\text{Total no. of blocks in cache}}{\text{No. of sets per block}}$$

$$\text{no. of block} = \frac{\text{cache size}}{\text{no. of word per block}}$$

" " $= \frac{2048}{4}$

no. of blocks in cache = 512

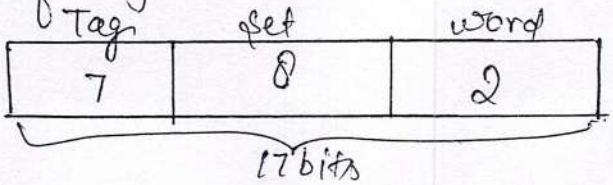
$\therefore$ no. of set $= \frac{512}{2}$

no. of set = 256

set bit $= 2^8 = 8$ bits

No. of Tag bit = 17 − (2+8)

No. of Tag bit = 7 bits



Ques:- A direct map cache has the following parameters.

Cache size = 1k words

block size = 128 words

main memory size = 64k words

Specify the no. of bits in tag, word field and block.

Sol^n:-

main memory size = 64k

$= 2^6 * 2^{10}$

$= 2^{16}$

MM bit = 16 bits

Cache size = 1k = 1024

block size = 128

no. of words = 128

$= 2^7$

word bits = 7 bits

$$\text{No. of blocks} = \frac{\text{cache size}}{\text{no. of word per block}}$$

$$\text{no. of blocks} = \frac{1024}{128}$$

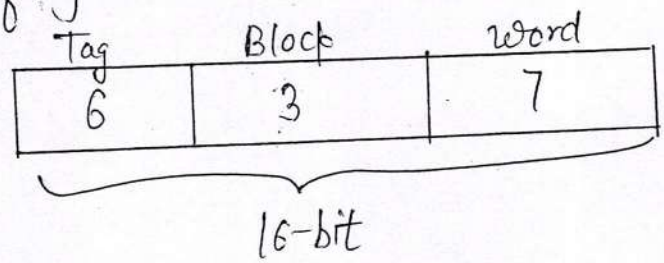
no. of blocks = 8

block bit = $8 = 2^3$

block bit = 3 bits

Tag bit = 16 - (7+3)

No. of Tag bits = 6 bit



Qus:- A computer system has 4k word cache organised in block set associative manner with 4 block per set, 64 words per block. The main memory size contains 65536 blocks. How many bits are there in each of the tag, set and word fields.

Sol'n:- main memory size = no. of blocks * no. of words per block

$= 65536 * 64$

$= 65k * 64$

$= 2^6 * 2^{10} * 2^6$

$= 2^{22}$

MM bit = 22 bit

$$\text{no. of blocks} = \frac{\text{Cache size}}{\text{no. of words per block}}$$

$$= \frac{4X1024}{64} \quad \frac{64}{16}$$

no. of blocks = 64

$= 2^6$

Block bits = 6 bits

No. of word bit = no. of words

$= 64$

$= 2^6$

word bit = 6 bits

$$\text{No. of sets} = \frac{\text{no. of blocks}}{\text{no. of set per block}}$$

$$= \frac{64}{4}$$

$= 16$

$= 2^4$

no. of set bits = 4 bits

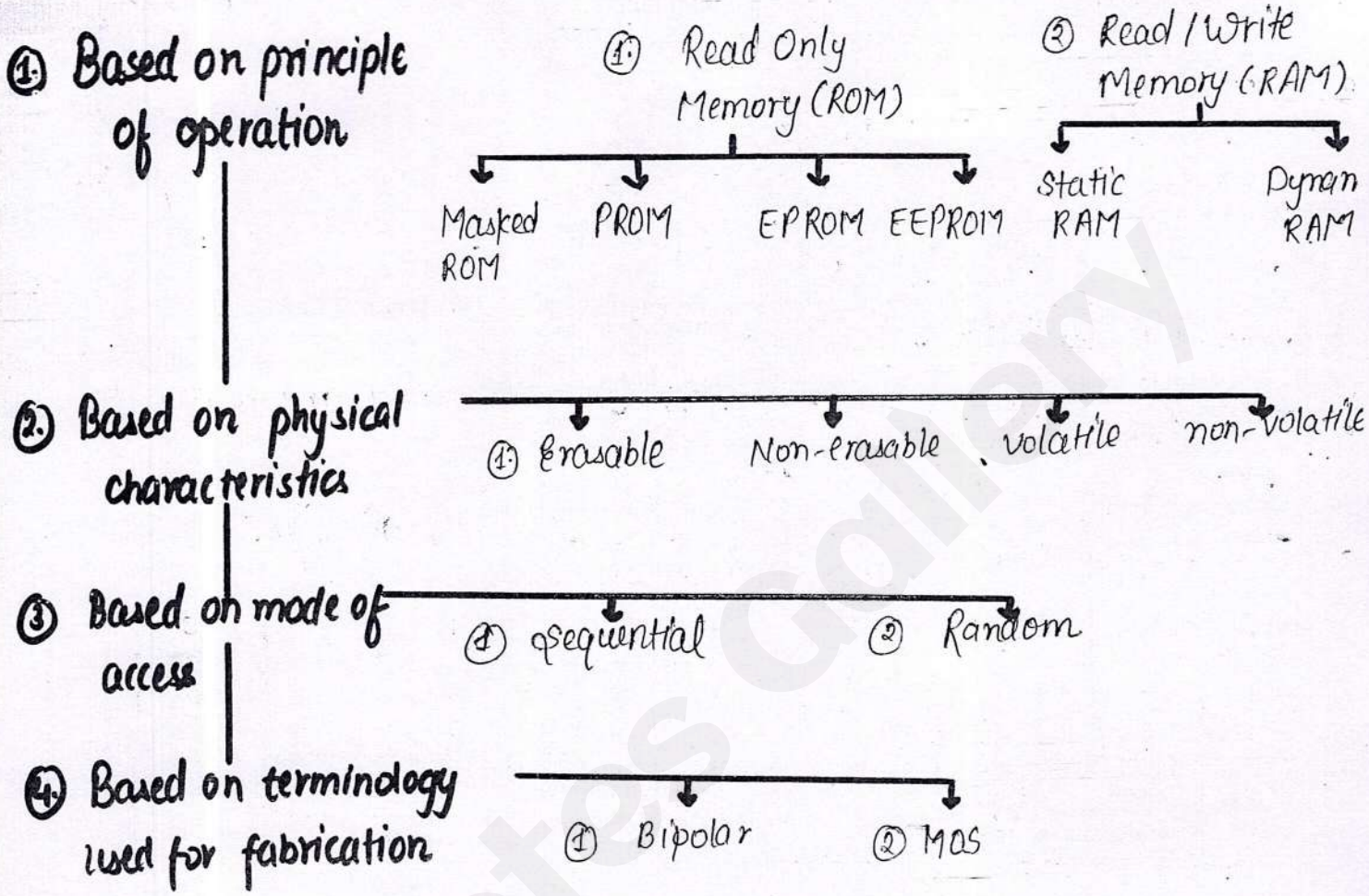
No. of Tag bits = 22 − (6+4)

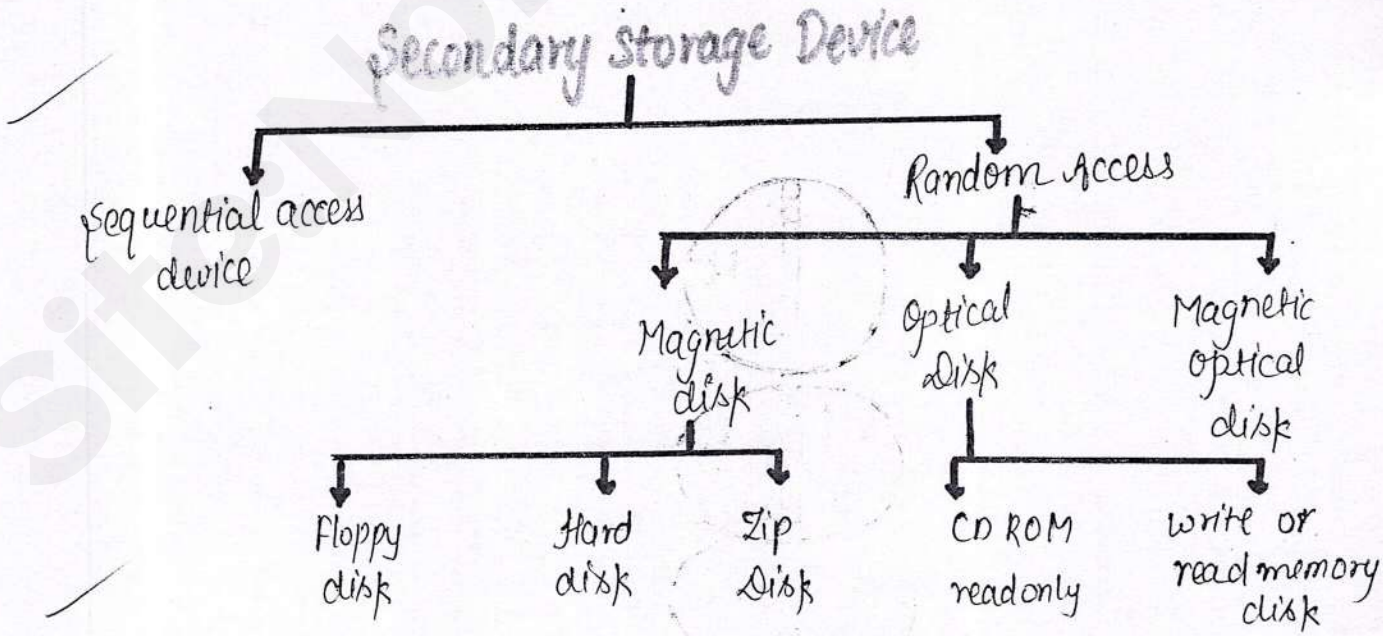
Tag = 12 bits

Tag	set	Word
12	4	6

22 bits

Classification of memory





Magnetic Disk:- It is an auxiliary memory that provides the bulk of secondary storage for modern computer system. A magnetic disk is a thin circular metal plate. It is coated with a thin magnetic film usually both sides.

- Digital information is stored on the magnetic disk by magnetizing the magnetic surface in a particular direction. Conceptually disk are relatively simple. Each disk plate has a flat circular shape like CD.
- Common plate diameter range is 1.8 to 5.25 inches. The two surfaces of platter covered with magnetic material.

Digital information can be stored on the magnetic field by applying current pulse of suitable to the magnetizing coil. The head are attached to disk arc that moves all the heads as unit.

- There may be thousands of cylinder in a disk drive and each track may contain hundreds of sectors.
- The storage capacity of common disk drive is in Gigabyte.

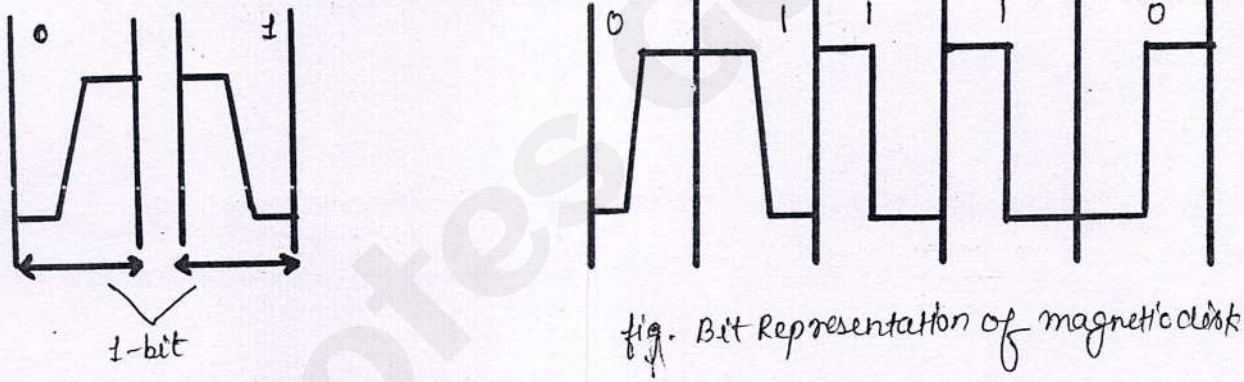
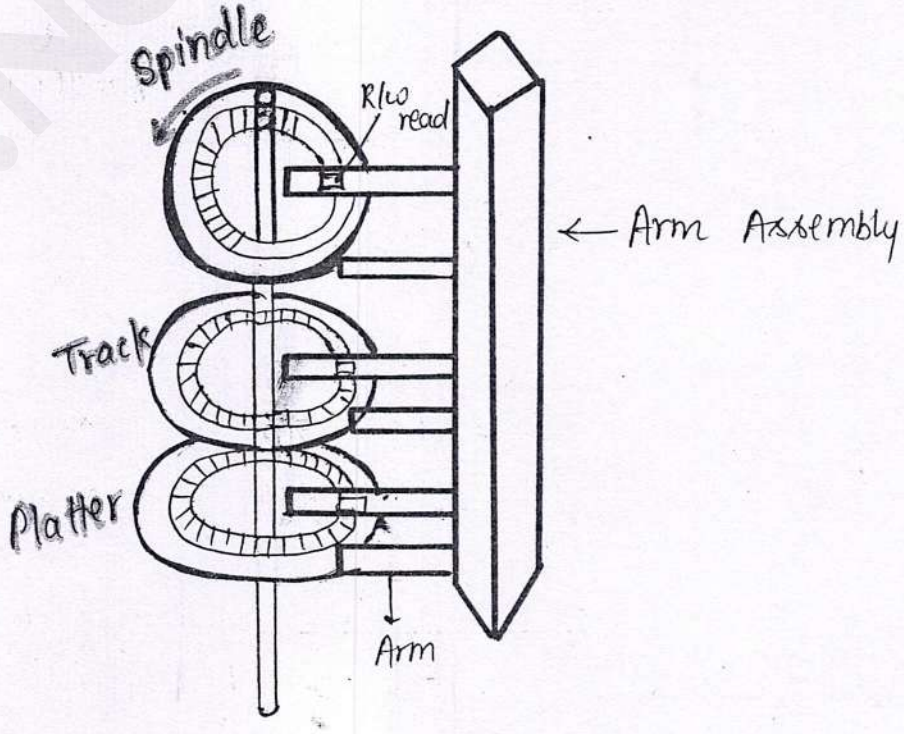
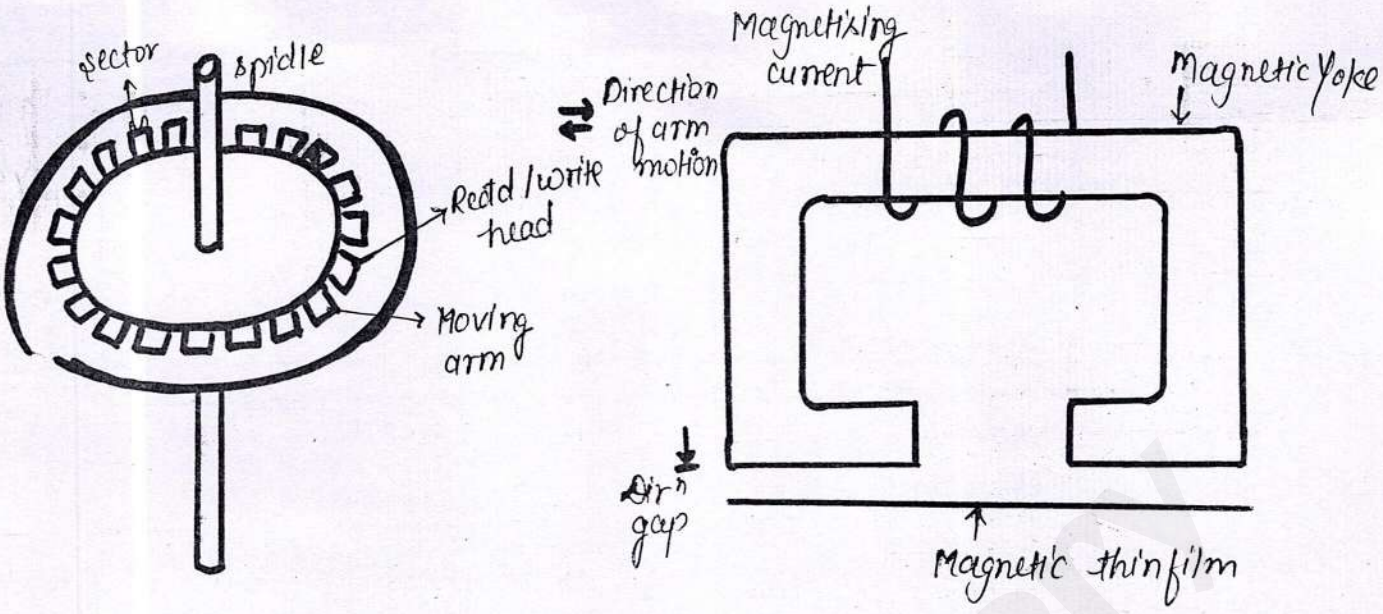


fig. Bit Representation of magnetic disk



∴ Mechanical structure of magnetic disk

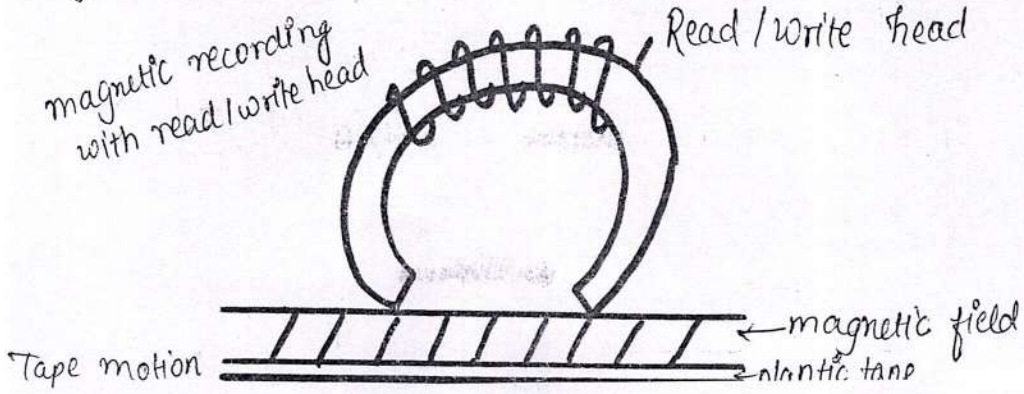


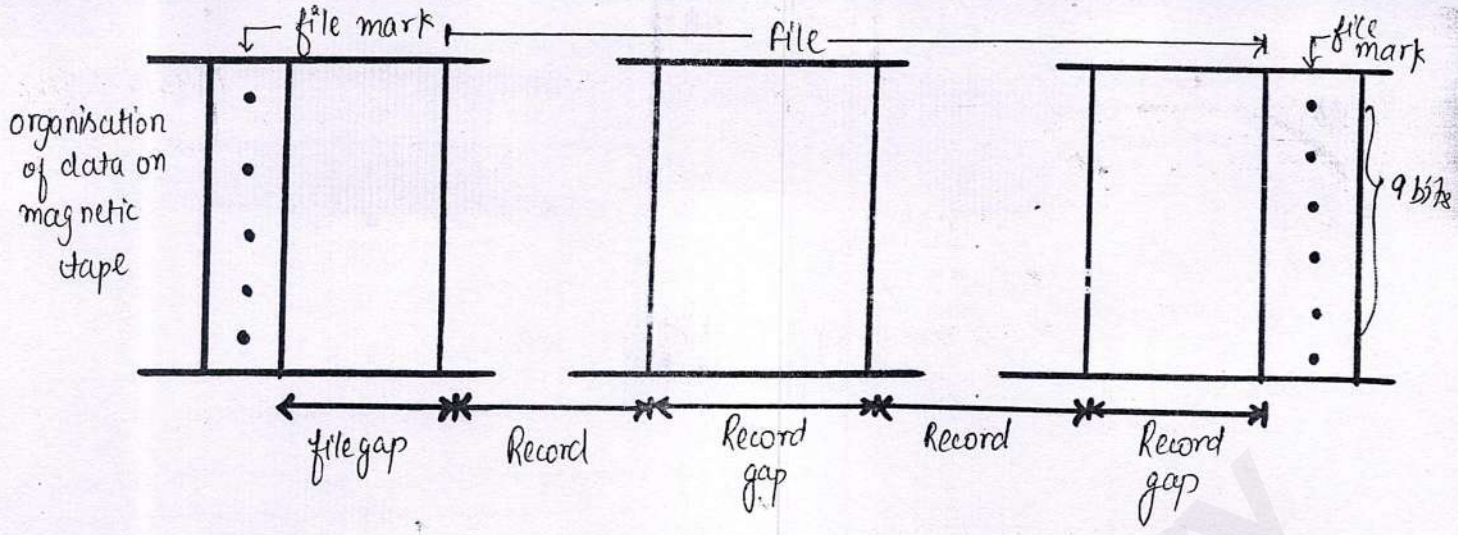
Magnetic Tape:- Magnetic tape is one of the most popular storage medium for large data that are sequentially accessed and processed. The tape is formed by depositing magnetic film on a very thin (1/2 inches or 1/4 inches) wide plastic tape. The tape ribbon itself is stored in reels similar to tape are automatically erased as new data for record in the same area.

The information is recorded on the tape with the help of read/write head. Although it is relatively permanent and can hold large quantity of data, its excess time in slow compare with main memory and magnetic disk.

In addition, random access to magnetic tape is about a 1000 times slower than the random access to magnetic disk, so tapes are not very useful for secondary storage.

Tapes are mainly used for backup for data which is not frequently used. They can store 20GB - 200GB data.





Optical Disk:- There are three basic types of optical disk :-

1. CD ROM
2. DVD
3. WORM (Blue-Ray) [Write once Read Many]

CD's can store 700 MB of data, DVD's can store upto 8.4 GB of data and Blue ray which is newest technology of optical media can store upto 50 GB of data.

This storage capacity is a clear advantage over floppy disk which only have the capacity of 1.44 MB.

Another advantage of that optical media over the profit is that it can used upto 7 times longer due to its durability

Virtual Memory:-

Virtual Memory technique that allow the execution of process that are not completely in main memory. The major advantage of this is, it can execute a program larger than memory as well as size of auxiliary memory.

When a program does not completely fit into memory, it is divided into segments. Segments which are currently being executed are kept in main memory and the remaining segment are stored in the secondary storage

If an executing program needs a segment which is not currently in the main memory, the required segment is copied from the ~~main memory~~ the req secondary storage (swap in). When a new segment of a program is to be copied from main memory it must replace another segment (swap out).

In virtual memory, the address that processor issues to access either instructions or data are called virtual & logical address virtual memory can be implemented by two methods,

1. Paging
2. Segmentation

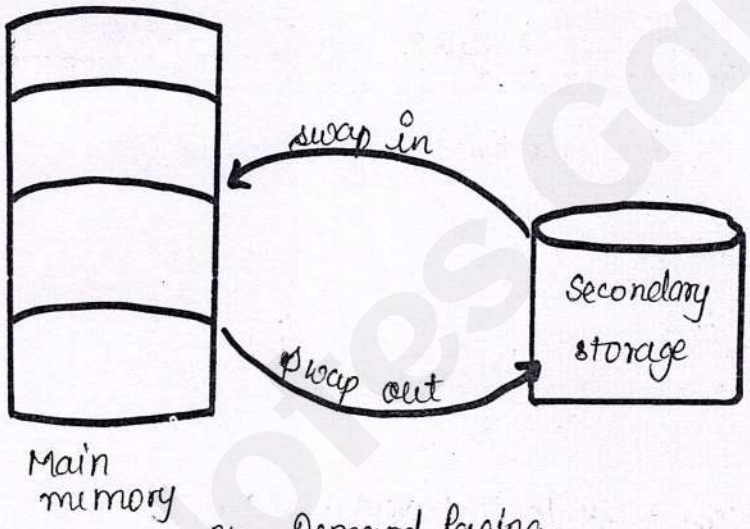


fig. Demand Paging

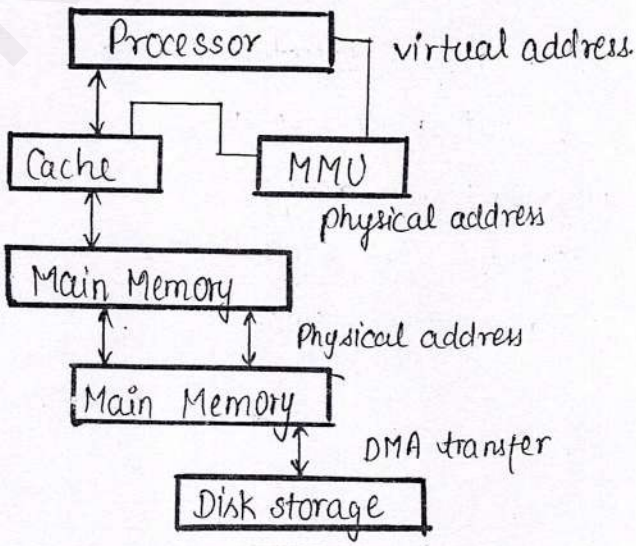
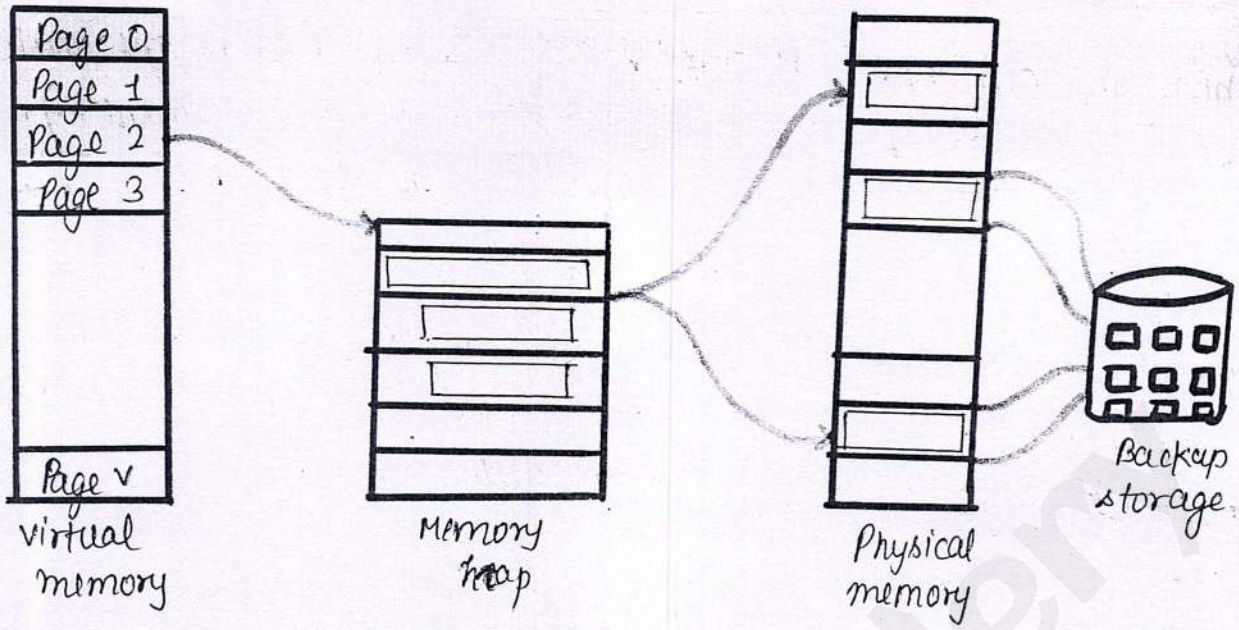


fig:- Virtual Memory Organization



:- Memory (VM) concept using paging

Address Translation in virtual Memory (Mapping process in virtual memory)

It involves two phases :-

1. Segment Translation
2. Page Translation.

Segment Translation :- A logical address consist of a selector and offset. A selector is the content of the segment register. Every segment selector has a linear base address associated with a segment descriptor. A selector is used to point a descriptor for the segment in description table. The linear base address from the descriptor is added to the offset to generate the linear address. This process is known as segmentation or segment translation.

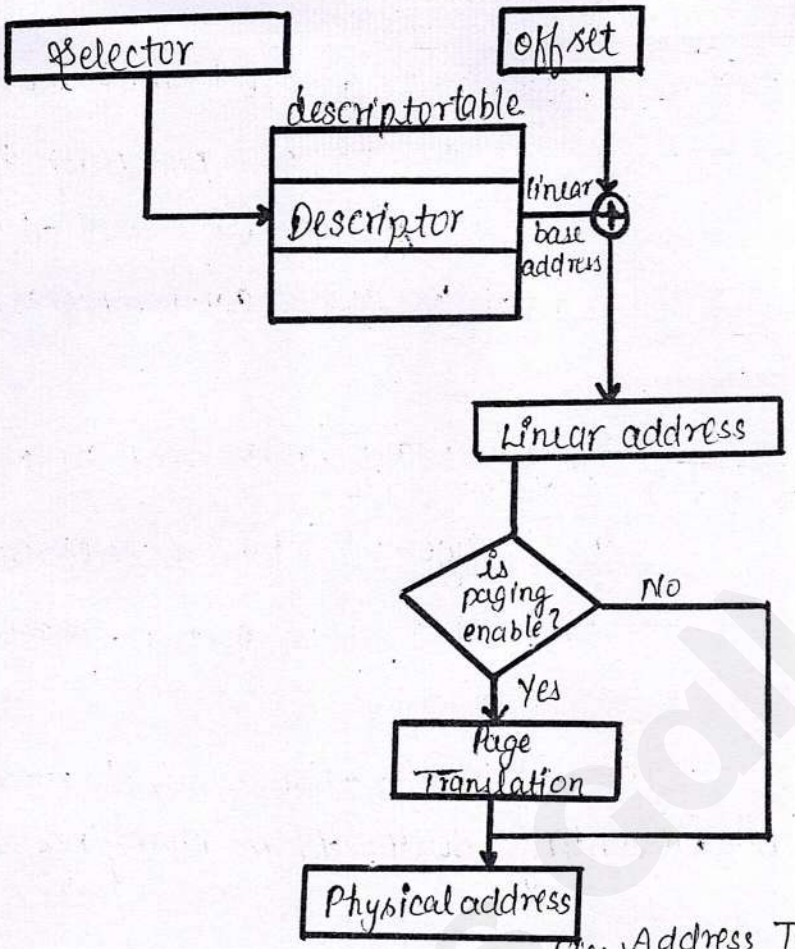
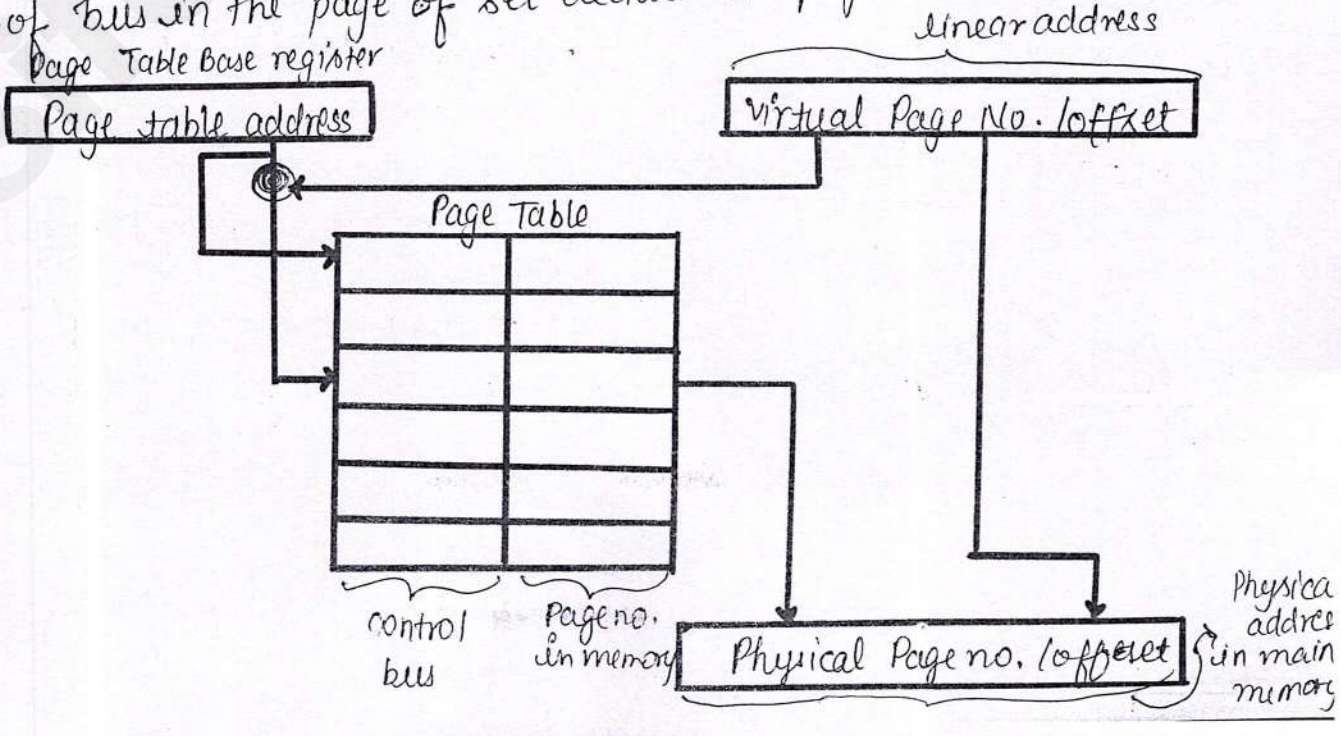


Fig: Address Translation.

Page Translation:- Page translation is the second phase of address translation, It transform a linear address generated by segment translation into physical address.

When paging is enabled, the linear address is broken into a virtual page no. and page offset. The physical page no. is not changed. The no. of bits in the page offset decides the page size.



Memory Management Hardware

A memory management system is a collection of hardware & software procedures for managing the various programs residing in the memory. Memory management system is a part of an operating system in computer. A collection of hardware components consists of memory management unit.

The basic component of memory management unit are :-

- A mechanism for dynamic storage allocation and deallocation
- A mechanism for sharing common program stored in memory by different user.
- A mechanism for protection of information against unauthorized access b/w user and preventing users from changing operating system functions.

Memory Protection

Memory protection is done by associating bits with the each page. These bits are kept in page table. A protection bit can define a page to be read/write, read only or hidden.

Protection bits can be checked to verify that number, now operation is done on read only. Every time a page in moved from one blok to another. It would be necessary to update the block protection bit.

Page Replacement Policies

Page Replacement Policies are strategies used by operating system to decide which page to remove from memory when there is a page fault. Common policies include-

FIFO (First In First Out)

LRU (Least Recently Used)

Optimal Page Replacement

(1.) **First In First Out (FIFO)**:- FIFO replaces the oldest page in the memory based on the order in which pages work were brought into the memory.

It uses a simple queue data structure to keep track of the order in which pages were loaded.

While easy to implement, FIFO may not always perform well in terms of page hit rate.

Belady Anomaly:- As we know that when we increase the frame size, the page fault decreases and page hit increases.

As per Belady's conclusion, in a FIFO algorithm for a certain sequence of memory references for a certain time period when we increase the frame size, the page fault also increase. This will become known as Belady's Anomaly. **Eg:-** Consider the following memory reference sequence 1,2,3,4,1,2,5,1,2,3,4,5. Explore that these reference string suffers from belady's anomaly.

Sol^n:-

	1	2	3	4	1	2	5	1	2	3	4	5
	1	1	1	4	4	4	5	5	5	5	5	5
		2	2	2	1	1	1	1	1	3	3	3
			3	3	3	2	2	2	2	2	4	4
								(H)	(H)			(H)

Framesize = 3

Page Hit = 3

Page fault = 12-3 = 9

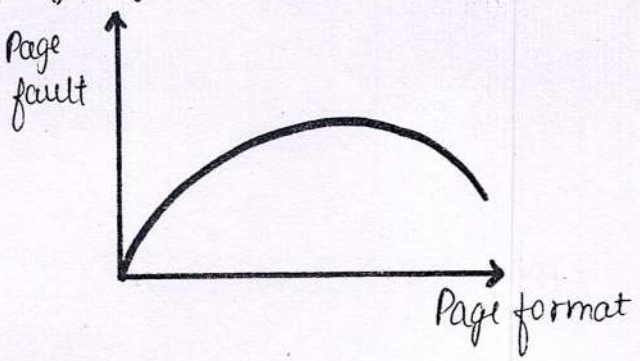
Frame size = 4

	1	2	3	4	1	2	5	1	2	3	4	5
	1	1	1	1	1	1	5	5	5	5	4	4
		2	2	2	2	2	2	1	1	1	1	5
			3	3	3	3	3	3	2	2	2	2
				4	4	4	4	4	4	3	3	3
					(H)	(H)						

Page hit = 2 Page fault = 10

Hence it suffers from Belady's anomaly because when we increase

frame size, page fault also increases.



(2.) **Least Recently Used (LRU):-** LRU replace the page that has not been used for the longest time.

It depends on the principle that pages that have not been used recently are less likely to be used in the near future.

(3.) **Optimal Page Replacement:-** Optimal Replacement selects the page that will not be used for the longest period in the future.

It serves as a theoretical benchmark as it requires knowledge of the future page accesses.

It never suffers from Belady's Anomaly.

Thrashing:- Thrashing in computer system refers to a situation where a computer's performance drops significantly due to excessive swapping of data blw the main memory and secondary storage

This occurs when the system have high degree of multiprogramming, causing it to spend more time moving data blw memory and storage then actually processing task. It leads to slow down in performance as a system struggles for memory resources.

Reasons of Thrashing

Thrashing occurs when a system is overloaded with too many tasks or processes, and the demand for physical memory exceeds the available RAM. This can happen due to :-

Insufficient Physical Memory :- If there is not enough RAM to handle the concurrent processes, the system has to depend on swapping data b/w RAM and the hard disk.

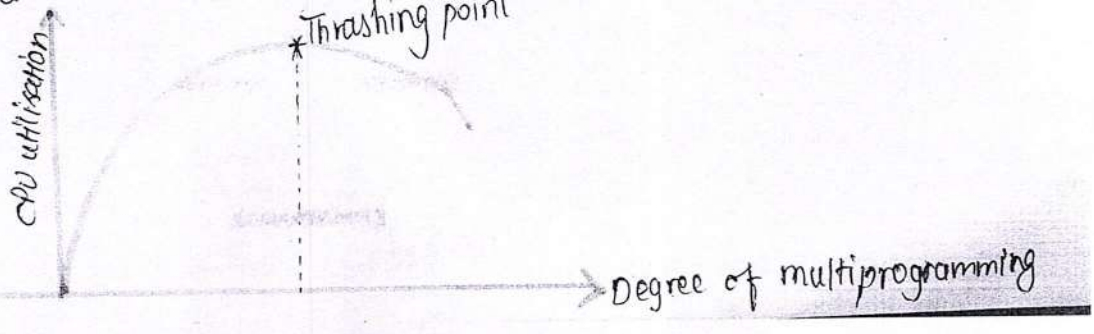
Inefficient Memory Management :- Poorly optimized memory management algorithms can contribute to thrashing. For eg :- If the system frequently swaps in and out large chunks of data unnecessarily, it can lead to performance degradation.

Core Overcommitting :- Allocating more virtual memory than system can practically handle result in thrashing. When the system tries to fulfill the excessive demand for virtual memory, it spends more time in swapping.

Solutions of Thrashing

To remove or prevent thrashing consider these solutions :-

1. Increase Physical Memory
2. Optimise memory uses
3. Use efficient paging algorithm
4. Adjust process priority.
5. Avoid overcommitting resources
6. Monitor and tune system performance
7. Consider solid state drives (SSD)



Associative Memory (Content Addressable Memory / CAM)

A memory unit accessed by the content is called an associative memory or content addressable memory (CAM).

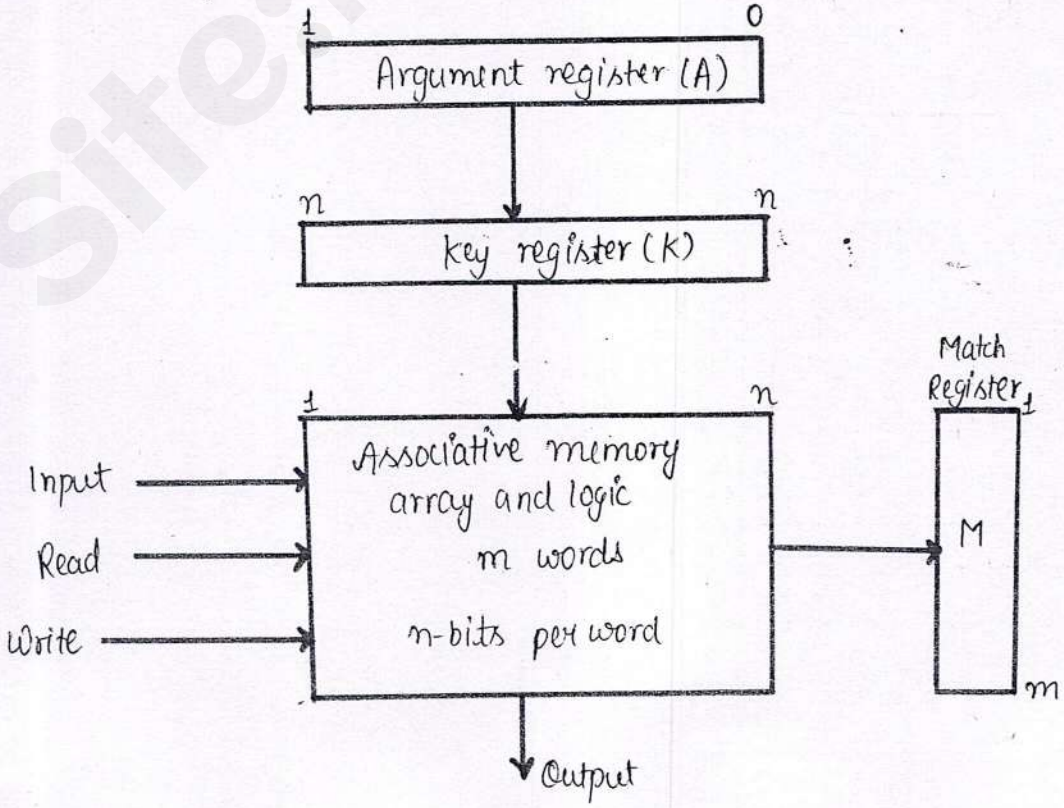
By using CAM, the time required to find an object in memory can be reduced.

In CAM, the memory is organised in such a way that the data itself serves as the lookup key.

This enables quick search and comparisons.

In a block diagram of an associative memory, consist of memory array with the match logic for n-bit words.

The argument register (A) and key register (K) each have n-bits per word. Each word in memory is compared in parallel with the content of argument register. The words that match with the word stored in argument register set a corresponding bits in the match register. Therefore, reading can be accomplished by sequential access to memory for those words whose corresponding bits in the match register have been set.



Locality of Reference :-

Locality of reference is a principle in memo management that suggest that when a prog. -am accesses a particular memory location, it is likely to access in the near future.

This principle is used for optimising the performance of computer syster When few procedures, line of codes that repeatedly call each other, these instructions are reside/localised in cache memory. This is referred to as locality of reference.

There are two main types of locality of reference :-

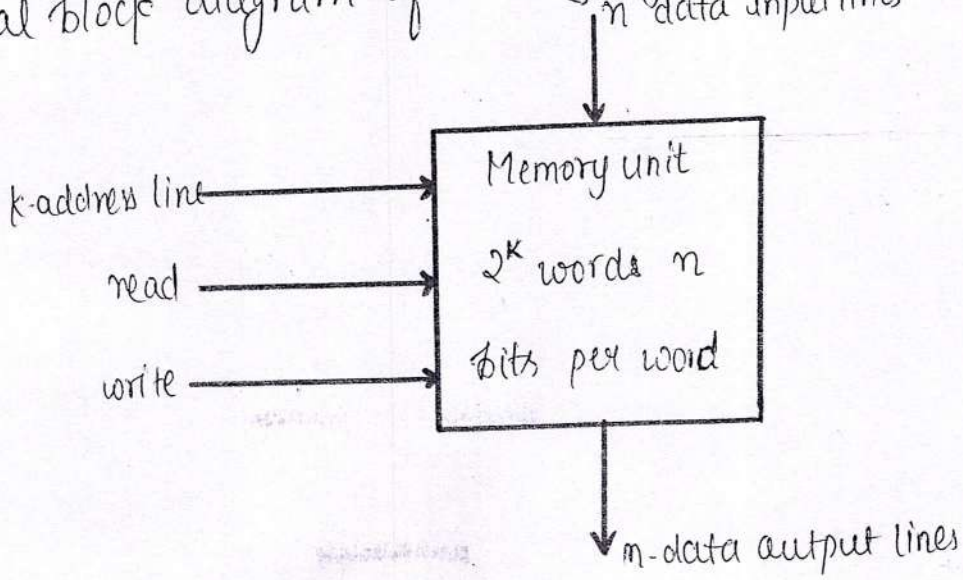
1. **Temporal :-** The temporal means that a recently executed instruction is likely to be executed again very soon. The tempora aspect of locality of reference suggests that whenever an instruction or data is first needed, it should be brought into the cache memory and it should remain there, until it is needed again.

2. **Spatial :-** The spatial means that instructions stored nearby the recently executed instructions are also likely to be execu -ted soon.

The spatial aspect suggests that instead of bringing just one instruction from memory to be caught, it is wise to bring many instructions that reside at adjacent address as well.

2-D (2-Dimensional) Memory Organization

The general block diagram of memory organization is :-



When there are k-address lines, then we can access 2^k words.

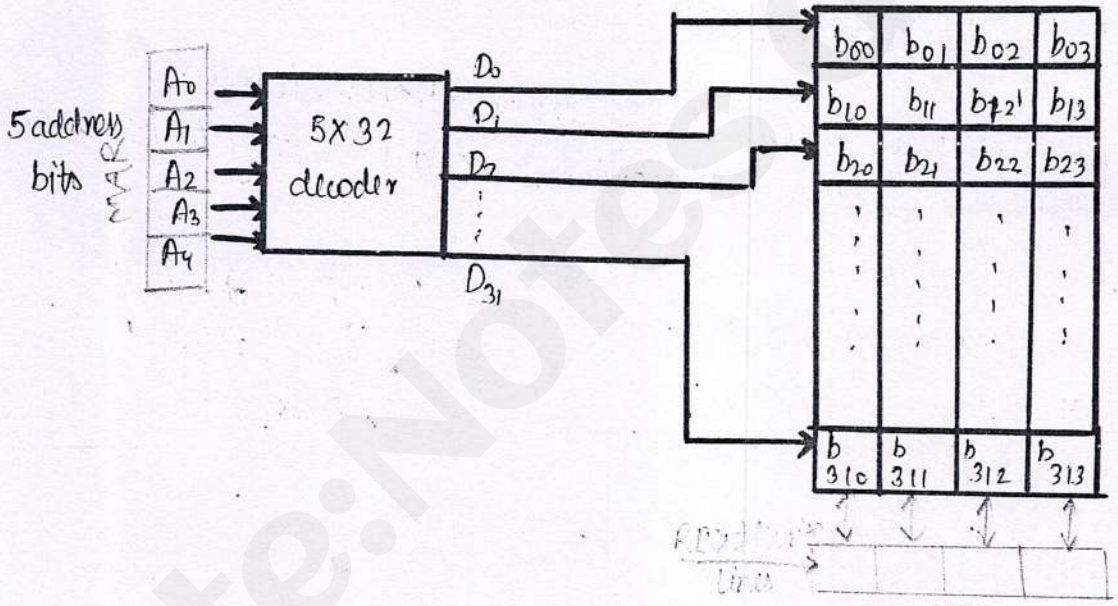
In 2D RAM organization a RAM of $2^n * m$ where 2^n memory words of m bits each.

It has a row select decoder of size $n * 2^n$ to select one out of 2^n words.

Each row (memory word) have m columns (one column for each individual bit).

In 2D RAM organization, hardware is fixed.

It requires more no. of logic gates. It is more complex because of large no. of wires and gates.



Here, we use 2 registers MAR and MDR. MAR holds the address of desired location from where we have to perform READ operation or we have to write memory word in memory.

MDR fetches all corresponding bits from memory in read operation and/or put the corresponding bits at particular address which is defined in MAR.

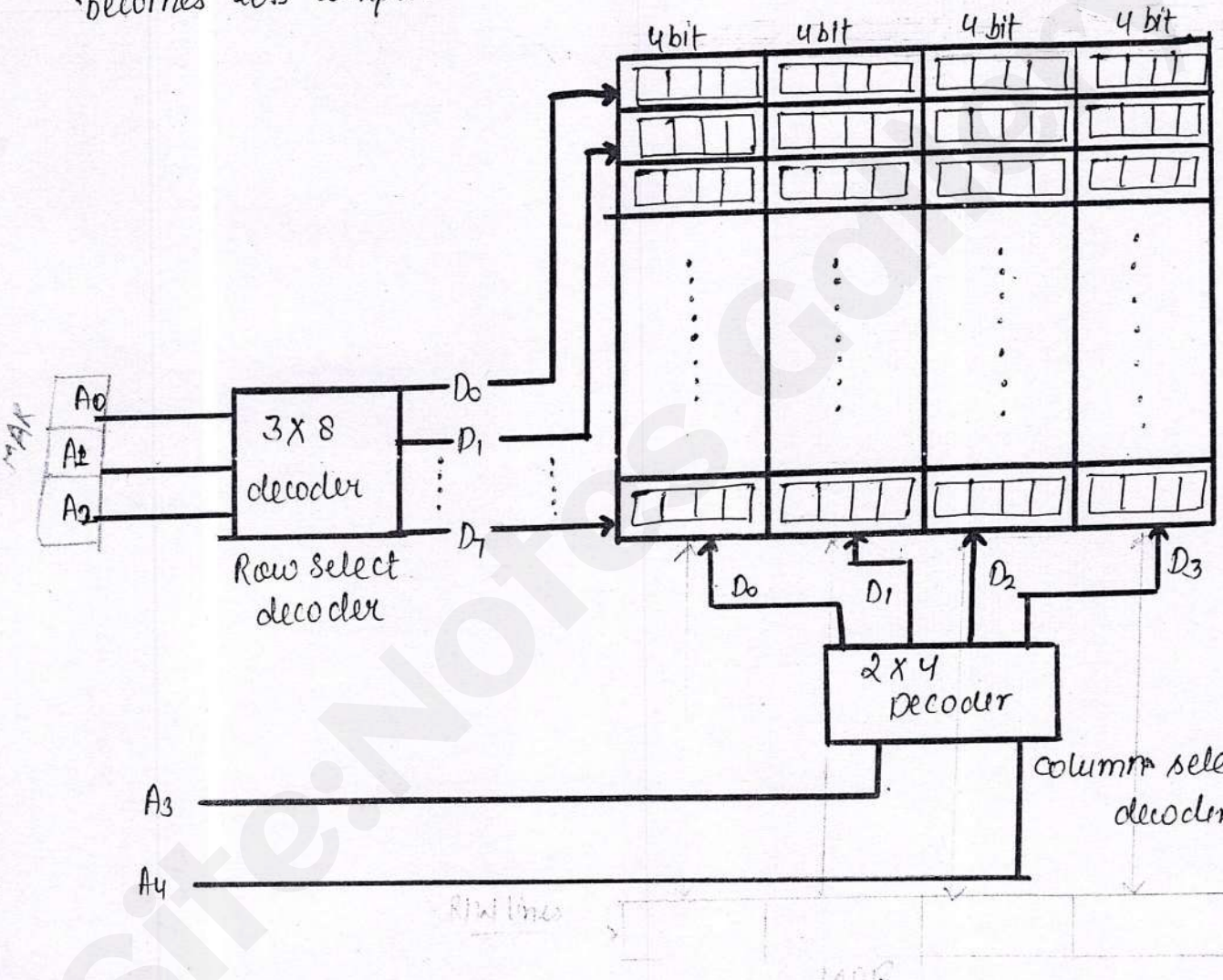
If we have to fetch one word from 1024 words then we have to use 10 X 1024 decoder which consists large no. of gates, circuit becomes complex. This drawback is reduced in 2.5/2½ RAM organization

$2\frac{1}{2}$ / 2.5-D Organization

In $2\frac{1}{2}$-D organization, the no. of address lines are divided into approximately equal size, one for row select and another for column select decoder.

In 2-D RAM, hardware is variable.

It requires less no. of logic gates. Because of less no. of gates circuit becomes less complex



Instead of using one decoder (5X32), we use two decoders, one for row selection and another one for column selection. In above diagram we used (3X8) decoder for row selection and 2X4 for column selection.

Let us consider, if we pass address line in row decoder 000, it enables first row of memory segment and 00 in column decoder then it activates first column, finally first word of memory layout will be activated.

Speed up and Amdhal's Law

Amdhal's law is used to calculate the performance gain that can be obtained by improving some portion of a computer.

It states that the performance improvement to be gain from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Speed up (performance improvement) tells us how much faster a task can be executed using the machine with the enhancement as compared to the original machine. It is defined as :-

$$\text{Speed up} = \frac{\text{performance for entire task using improved machine}}{\text{performance for entire task using original machine}}$$

$$\text{Speed up} = \frac{1}{(1-Fe) + \frac{Fe}{Se}}$$

where Fe = Fraction enhance

Se = Speed up enhance