COA Unit 4 Notes

Minakshi.

Computer Memory:

→ The memory unit is an essential component of any digital computer since it stores the program and data.

Key characteristics of computer Memory systems:

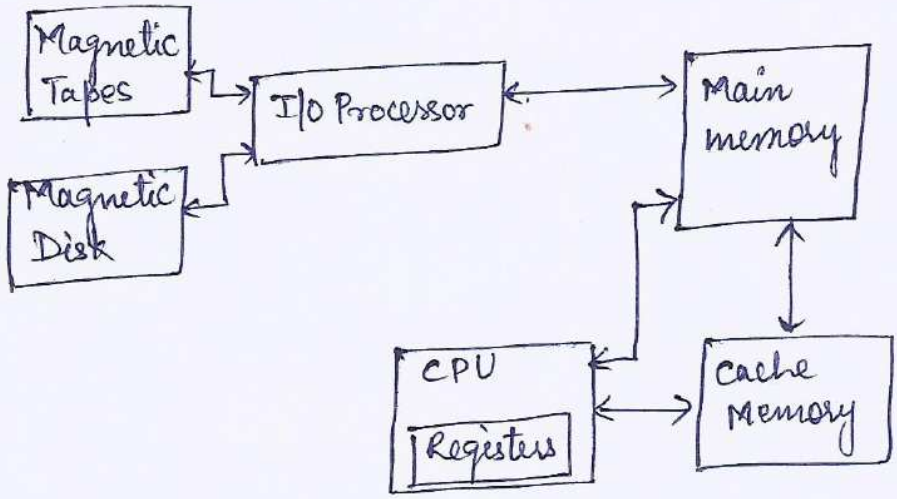
1. **Location**
   - Processor
   - Internal (Main memory)
   - External (Secondary)
2. **Capacity**
   - word size
   - Number of words
3. **Unit of Transfer**
   - Word
   - Block
4. **Access Method**
   - Sequential
   - Direct
   - Random
   - Associative
5. **Performance**
   - Access time
   - Cycle time
   - Transfer rate
6. **Physical Type**
   - Semiconductor
   - Magnetic
   - optical
   - Magneto-optical
7. **Physical characteristics**
   - Volatile / non volatile
   - Erasable / non-erasable
8. **Organization**

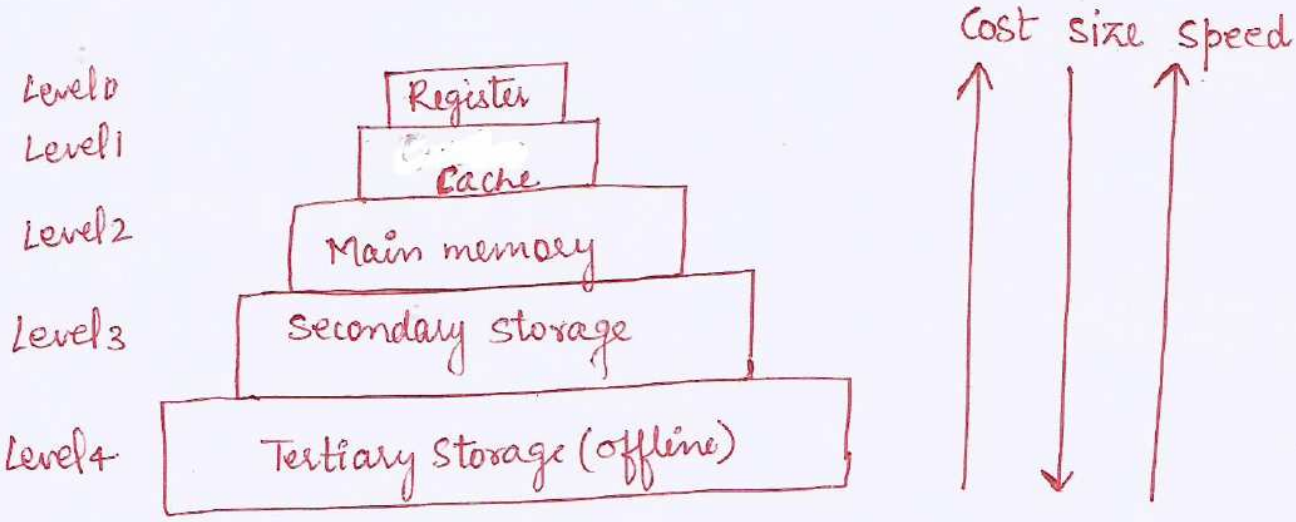
②

Memory Hierarchy:

→ The memories used in computer system are organized in a hierarchy.

Memory Hierarchy in computer System





Registers:

→ Registers are the part of CPU
→ fastest
→ smallest
→ costly
→ implemented using Flip Flops

Cache Memory:

- cache memory is placed between CPU and main memory.
- Faster than Main memory.
- → Most recently used information is stored in cache.
- → cache memory is implemented using static RAM.

Main Memory:

- Main memory can directly communicate to the CPU.
- The information is first kept in the main memory to get accessed by the CPU.
- Main Memory is implemented using Dynamic RAM (DRAM)

Secondary Storage: (External Memory)

Level 3: Secondary storage like magnetic disk are present.

- Non-volatile type of memory.
- used as backup storage.
- cheaper and larger than main memory.

Level 4: (Removable Media or offline media)

- The data not being currently used can be kept in removable media. Eg. CD, Pendrive etc.
- Non-volatile
- cheaper
- Larger in size.

Main memory: (Primary Memory or simply memory)

- Semiconductor memory and internal memory.
- can directly communicate to the processor and auxilliary memory with the help of I/O processor.

- Two types of main memory
  - → ROM (Read only Memory) ⇒ Non volatile
  - → RAM (Random Access Memory) ⇒ Volatile

- Data transfer b/w main memory and CPU takes place by.
  - MAR ⇒ Memory Address Register
  - MDR ⇒ Memory Data Register

Organization: The basic element of semiconductor memory is 'Memory cell.'

- Memory cell represents two states zero and one.
- Memory cell is capable of being written into (at least once), to set the state
- Memory cell is capable of being read to sense the state.

Writing Reading

Types of RAM:

→ Static RAM
→ Dynamic RAM

Static RAM:

→ Faster than DRAM
→ Expensive
→ Used for cache memory.
→ single block of memory needs 6 transistors
→ complex uses transistors and latches.

DRAM:

→ Slower than SRAM
→ cheaper than SRAM.
→ used in Main memory
→ simple and uses capacitors and very few transistors
→ stores charges on capacitors
→ needs periodic charging (or referenshing)

ROM:

- ROM stands for Read only Memory. ~~The mem~~
- we can only read from this type of memory.
- information is stored in such memories during manufacturing
→ used as Bootstrap (a program to ~~to~~ start computer).

Types of ROM

→ PROM
→ EPROM
→ EEPROM
→ Flash Memory

PROM (Programmable ROM):

→ can be modified only once by a user.
→ it is not erasable.

EPROM: (Erasable and Programmable ROM)

→ can be altered or modified multiple times
→ updated each time from initial stage.
→ Data erased completely.
→ More expensive than PROM.

EEPROM: (Electrically Erasable Programmable ROM)

→ can be altered or modified multiple times
→ selectively programmed and erased
→ More flexible than EPROM.
→ More expensive than EPROM.

Flash Memory:

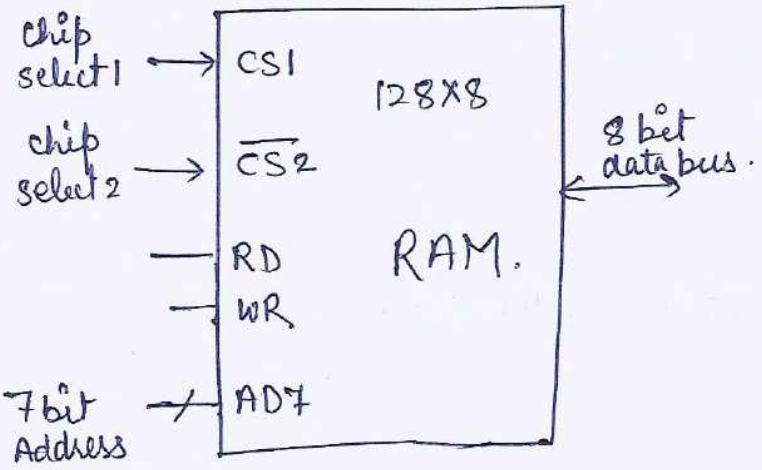
→ is a type of EEPROM.
→ is programmed and erased in large blocks
→ may also be a standalone memory storage device ~~dev~~ such as a USB drive.
→ can be put between EPROM and EEPROM in both the cost and functionality.

EPROM → entire chip erasable.

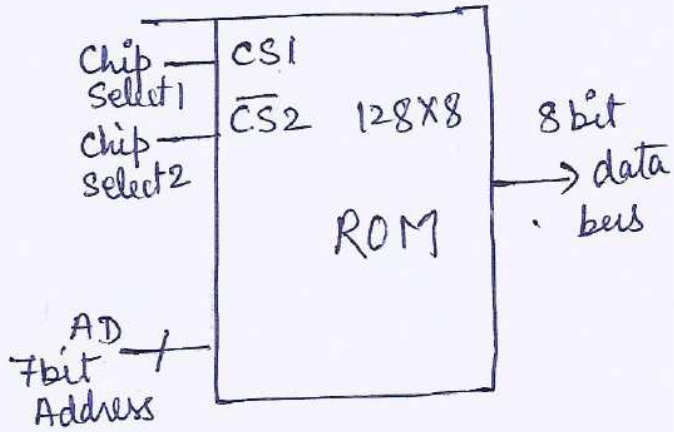
Flash → block erasable

EEPROM → byte erasable

RAM chip:



ROM chip



RAM size ⇒ m X n ← ~~word~~ size of each word.

↑ denotes number of words in RAM.

Ques: if RAM chip size is 128X8. then how manny RAM chip will be used to make 2048 bytes of RAM.

RAM chip size = 128X8

RAM size required = 2048X8 because 2048 bytes

Number of chips required $= \frac{2048}{128} = \frac{2^{11}}{2^{7}} \Rightarrow 2^{4} = 16$ chips

The decoder to select 16 chips

4X16 ⇒ size of decoder required to select one of the 16 chips.

Ques:

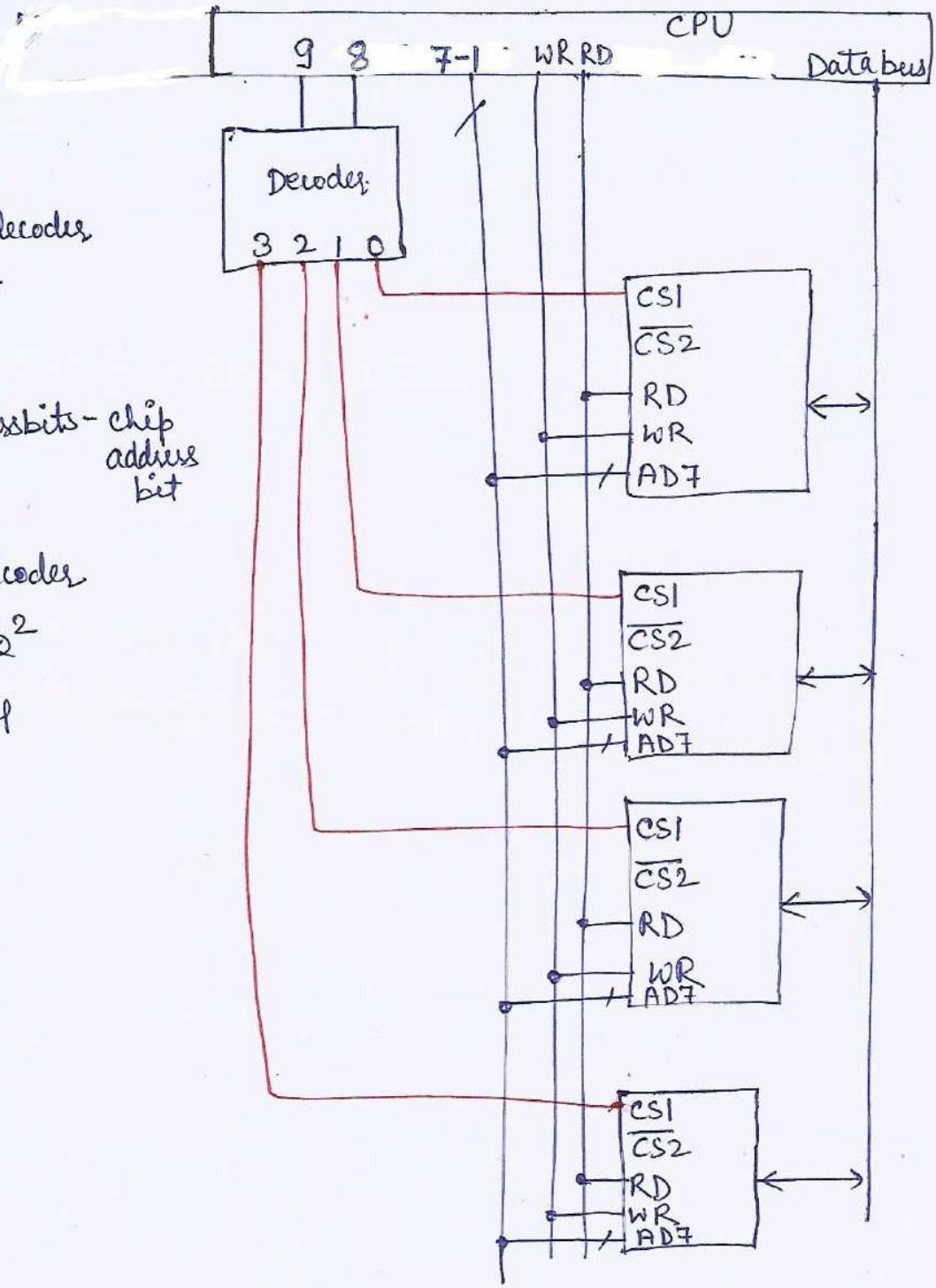
RAM chip size = 128X8

Memory size required 512X8

Number of chips required ⇒ $\frac{512}{128}$ ⇒ 4

Address lines in RAM chip ⇒ 7

Address bits for memory ⇒ 9



inputs of decoder

= 9-7

= 2

⇒ Addressbits - chip address bit

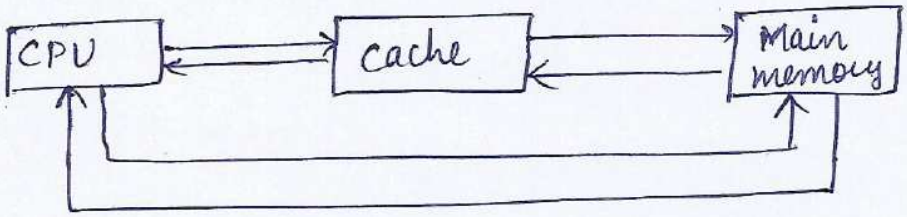
Size of decoder

= 2×2^2

= 2X4

Cache Memory :

- cache memory is an internal memory placed between CPU and main memory.
- Most recently used information is kept in cache memory.



* The cache contains the copy of some portions of main memory when the CPU tries to read a word from main memory, cache checked is performed, if that word is present in cache or not.

  if yes, the word is delivered to CPU

  if no, a block of memory consisting of that word is read into the cache and is delivered to the CPU.

Imp

* Also, cache memory satisfies 'Locality of Reference'. That is the block of data which is fetched from the main memory is likely that there will be more future reference for that block of word.

→ Temporal Locality of Reference

→ Spatial Locality of Reference.

Cache Fundamentals :

Cache Hit : when data is found in cache.

Cache Miss : when data is not found in cache.

Hit time : Time taken to access the cache. (CPU ← cache)

Miss Penalty : Time taken to move data from main memory to cache and then to CPU. when ~~word is~~ data is not found in cache
(CPU ← cache ← main memory)

Hit Ratio : The hit ratio is the fraction of accesses which are a hit

Miss Ratio : The miss ratio is the fraction of accesses which are a miss.

Structure of cache / Main memory :

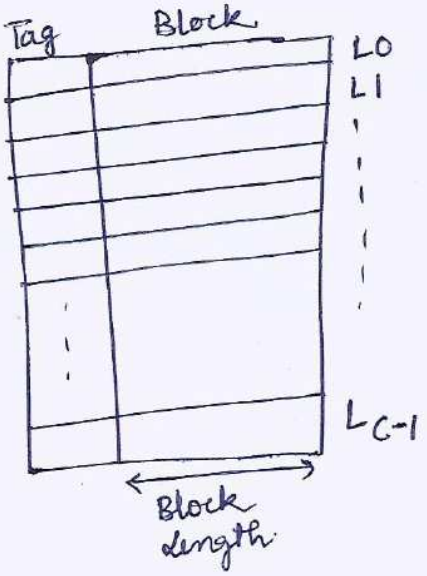
The main memory has 2^n addressable words with n-bit addresses each

For mapping purpose, main memory is devided into fixed length blocks with X-words each

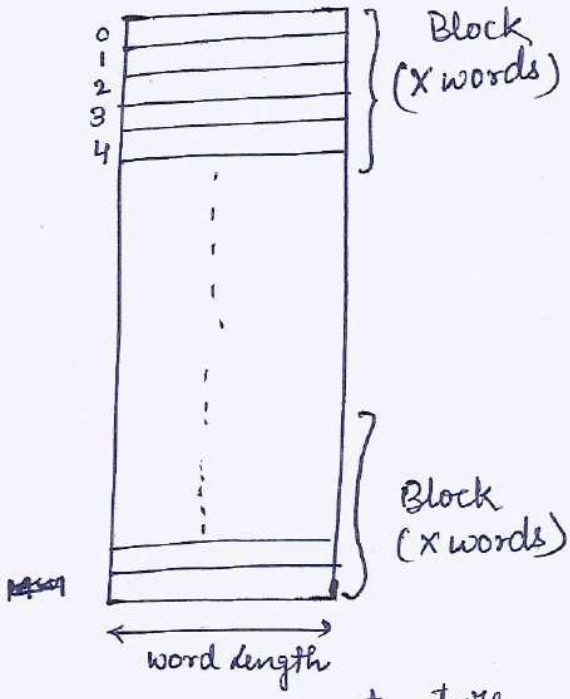
$$\text{Number of blocks } M = \frac{2^n}{X} \text{ blocks} = \frac{2^n \text{ words in main Memory}}{X \text{ number of words in each block.}}$$

- cache consists of C lines of X words each
- Each line consists of X words.

$$\boxed{C << M}$$



C ⇒ Number of lines in cache.
M ⇒ Number of blocks in MM



Main memory structure

When a block of memory is read, it is transferred to one ~~one~~ of the lines of cache. Because there are more blocks than lines, one line can not be assigned permanently to one particular block. So each cache line has 'tag' that identifies which particular block is being stored.

Cache Design issues:

Element of Cache Design

- Cache parameters
  - → Block size
  - → Line Size
  - → Cache Size
- Mapping function
  - → Direct
  - → Associative
  - → Set Associative
- Replacement Algo
  - FIFO
  - LRU
  - optimal
- Write Policy
  - Write Through
  - Write Back
  - Write Once

Cache Parameters:

- The size of the cache should be small enough, so that the cost per bit is less but the size should be large enough.
- The larger block size reduce the number of blocks that fits into the cache, so block size should not be large.

Mapping Function:

There are fewer cache lines than memory blocks, an algorithm is required ~~for~~ for mapping of memory blocks into cache lines.

* Mapping function: maps the memory block into a cache line

Types of mapping function: cache mapping techniques

→ Direct
→ Associative
→ Set Associative.

Address Mapping or Cache Mapping

- The size of memory is smaller than main memory. Therefore, There should be some mechanism to fill the data in the cache. Here the mapping techniques can be used.
- Address mapping is mapping of main memory blocks into cache lines
- Three techniques:
  - → Direct Mapping
  - → Associative mapping
  - → Set-Associative Mapping

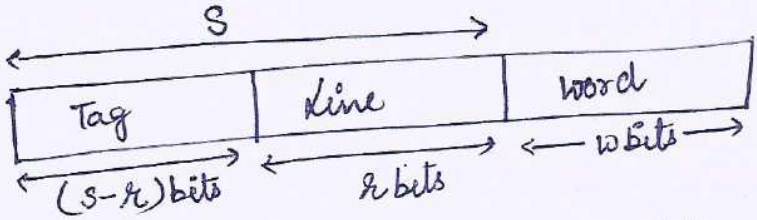
Direct Mapping:

- In this mapping, each block of main memory maps into only one possible cache line.
- It can be expressed as

$$\boxed{i = j \bmod m}$$

where $i \rightarrow$ cache line number
$j \Rightarrow$ Main Memory block number
$m \Rightarrow$ number of cache lines.

- Each main memory address can be divided into three parts



Tag ⇒ tells which block is currently being stored in cache
Line ⇒ line bits identifies one of the cache lines ($m = 2^r$)
$m \Rightarrow$ number of lines $\Rightarrow 2^r$

word bit or Block offset bit identifies which word (or byte in case of bytes ~~in block~~) in the block.

~~S → address length.~~

Address Length = $(S+W)$ bits

Number of words or addressable unit = 2^{S+W} words or bytes

Block size = Line size = 2^{W} words or bytes

Number of blocks in main memory = $\frac{2^{S+W}}{2^{W}} \Rightarrow \frac{\text{Total number of words in main memory}}{\text{Block size}}$

$= 2^{S}$

Number of bits to identify the blocks in main memory = S bits = Block index

Number of lines in cache = $\frac{\text{Total number of ~~lines~~ words in cache}}{\text{Line size}} = 2^{r}$

Number of bits to identify lines = r = cache index

Tag bits = Block index − cache index

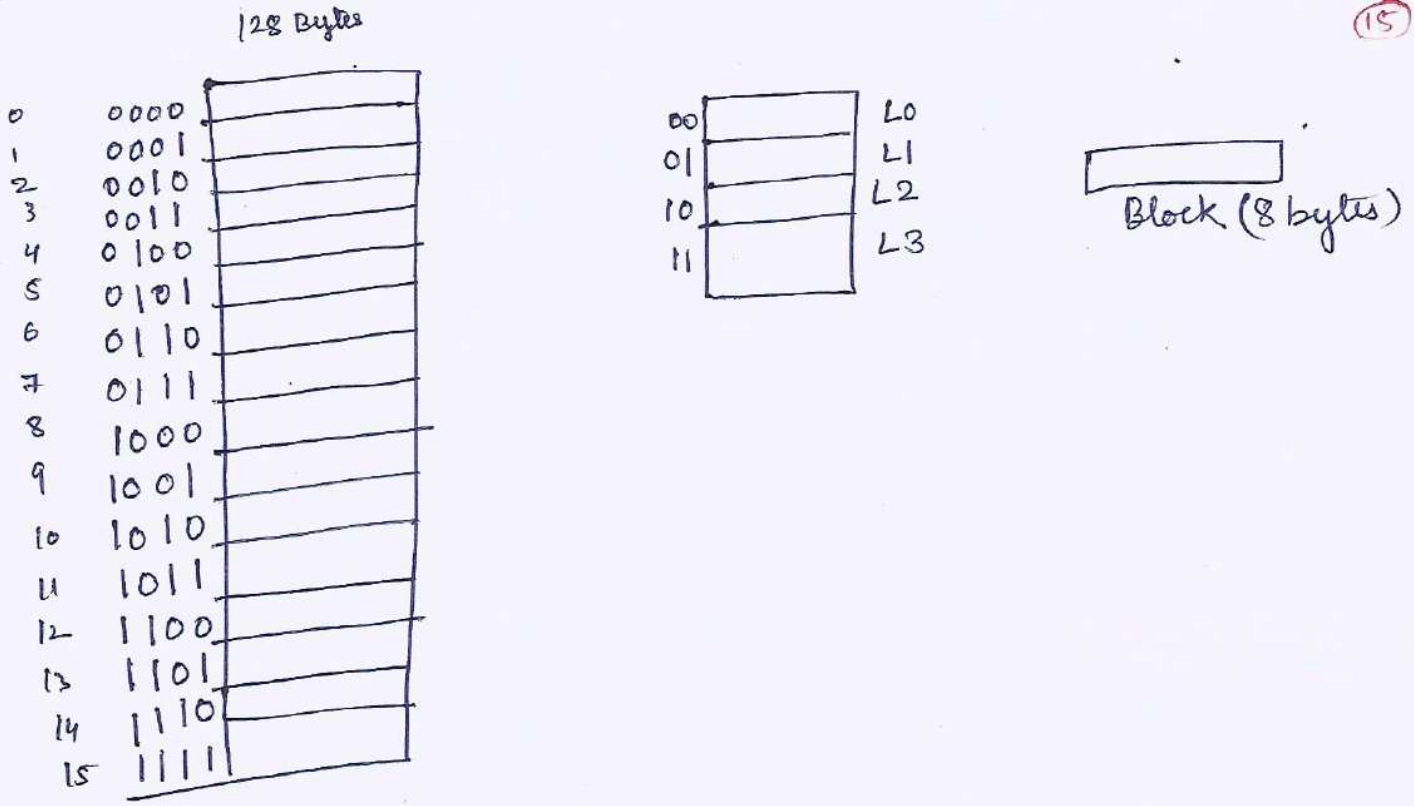
$= (S-r)$ bits

<u>Example</u> <u>Direct Mapping</u>

Main memory size = 128 bytes

Cache size = 32 bytes

Block size = 8 bytes



(BI) Block index ⇒ 4 bits

(CI) cache index = 2 bits

Formula ⇒ $\boxed{i = j \bmod m}$

$i \rightarrow$ line number.

$j \rightarrow$ block number.

$m \rightarrow$ number of lines in cache.

$m = 4$

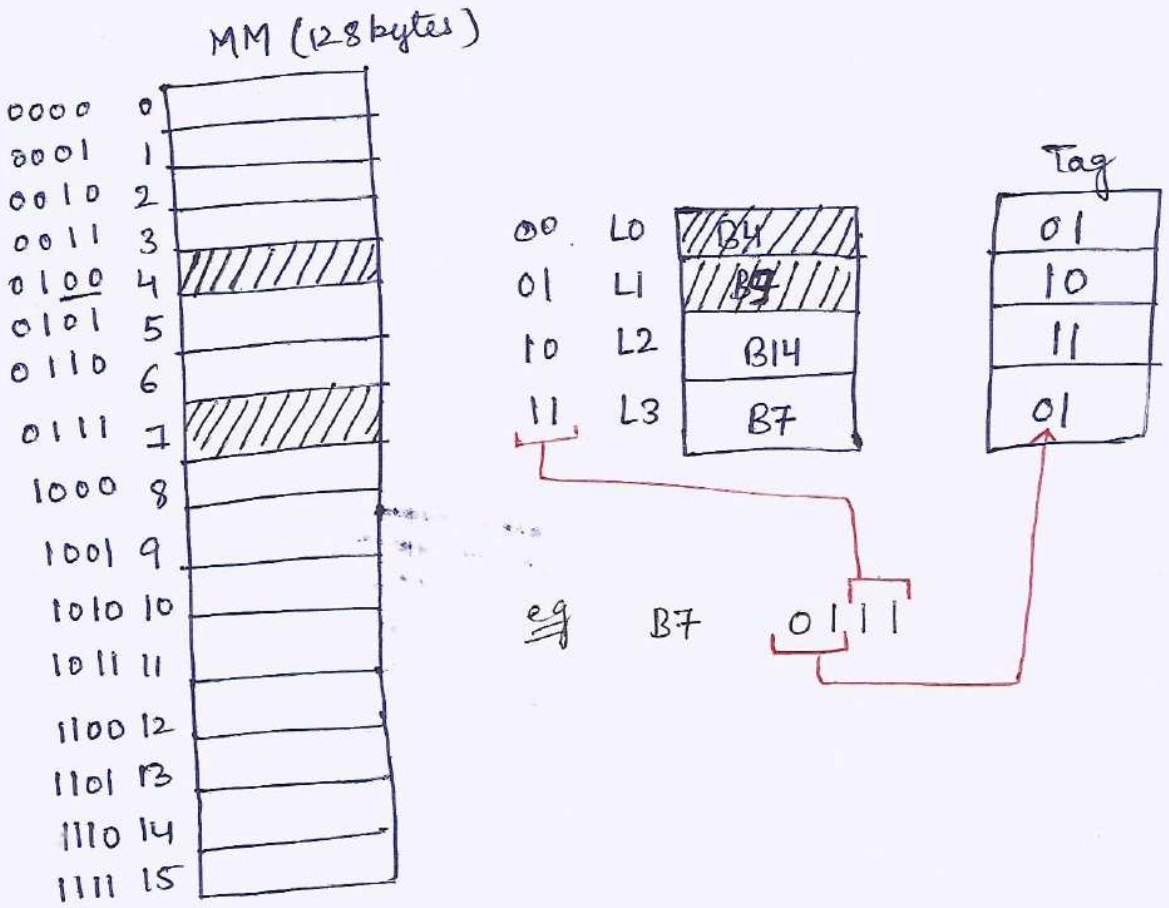
Line number	Blocks can be placed
L0	B0, B4, B8, B12
L1	B1 B5 B9 B13
L2	B2 B6 B10 B14
L3	B3 B7 B11 B15

* The least significant Block index bits are matched with cache index bits

So line 0 can hold B0, B4, B8 and B12. So CPU to understand which block in the cache, it maintains some extra information bit which is called tag bits.

→ So for each line CPU holds tag bits

* In direct mapping Tag holds the (BI-CI) ~~bits~~ MSB bits of the block index BI.
  - Tag holds the Most significant bits of the block.



Disadvantage of Direct mapping:

→ Direct mapping imposes restriction on the placement of a particular memory block into a particular cache line.

→ Poor hit ratio.

Advantage: Simple to implement.

- Number of blocks in main memory $= \dfrac{\text{Main memory size}}{\text{Block size}}$

$$= \frac{128 \text{ bytes}}{8 \text{ bytes}} = 16 \text{ blocks.}$$

- Number of lines in cache $= \dfrac{\text{Cache memory size}}{\text{Block size}}$

$$= \frac{32 \text{ bytes}}{8 \text{ bytes}} = 4$$

- Block index bits $= \log_2(\text{Number of blocks})$

$$= \log_2 16 = \log_2 2^4 \Rightarrow 4$$

Block index $= 4$ bits

- Cache index bits $= \log_2(\text{Number of lines})$

$$= \log_2 4 = \log_2 2^2$$
$$= 2\log_2 2$$
$$= 2$$

- Tag bits $=$ Block index $-$ cache index

$$= 4 - 2$$
$$= 2 \text{ bits}$$

- Address bits $= \log_2(\text{Main memory words})$

$$= \log_2(128)$$
$$= 7 \text{ bits}$$

Associative Mapping: or Fully Associative Mapping:

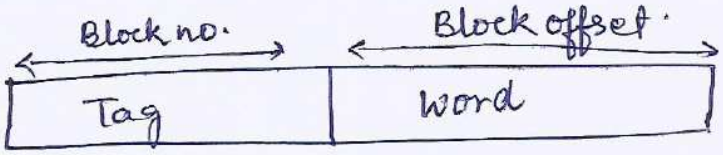
→ Overcomes the problem of direct mapping method.

→ In associative mapping any main memory block can be placed in any cache line.

→ whenever a main memory block is copied into cache line, the complete block index is placed in the tag bits.

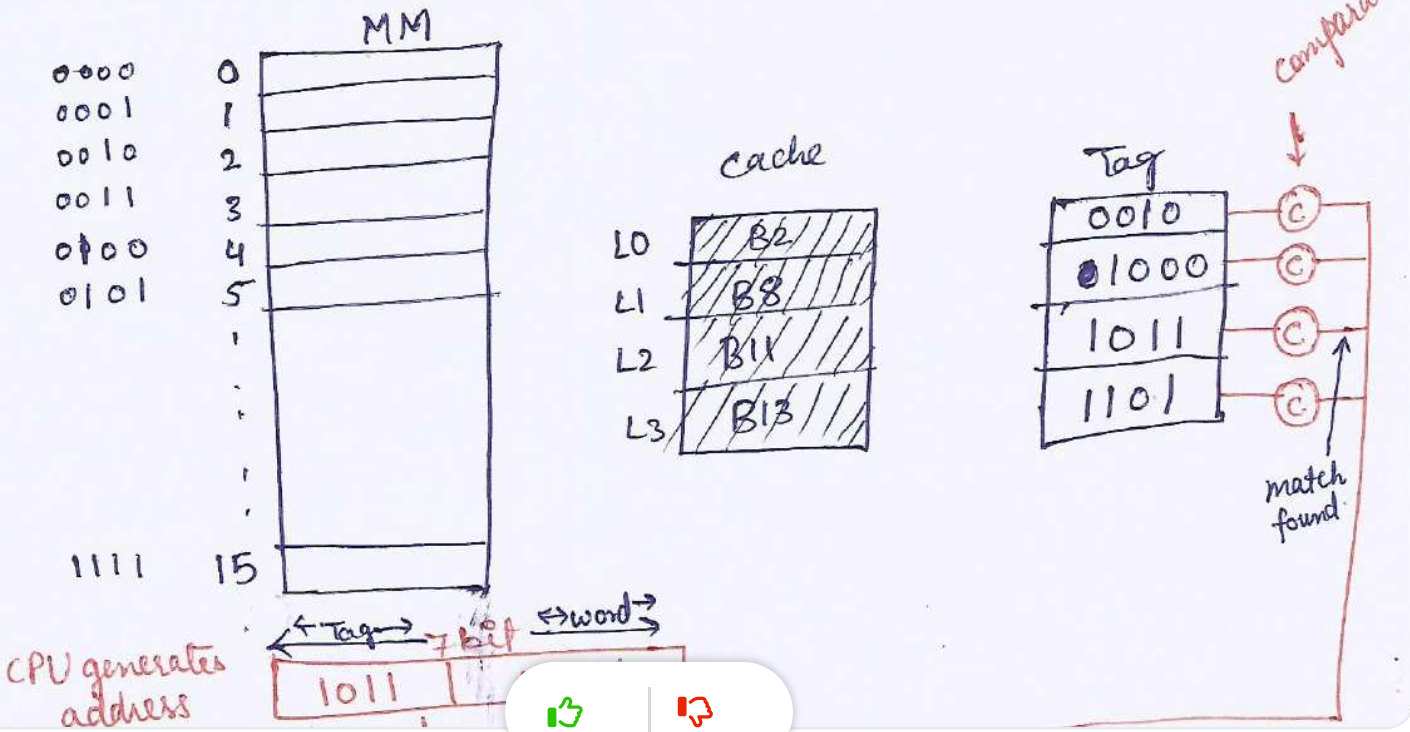
Tagbits = Block index bits

→ Address bits can be divided into two parts



Example Main memory size = 128 bytes
Cache size = 32 bytes
Block size = 8 bytes.

⇒ Number of blocks = 16 BI = 4
⇒ Number of lines = 4 CI = 2



Advantages:

→ Very High Hit ratio

→ better block placement strategy which ~~reduced~~ reduced contention.

Disadvantages:

→ Associative memory is costly.

→ Significant increase in tag length

→ complex :

→ Searching is an overhead.

Set Associative ~~Memory~~ Mapping

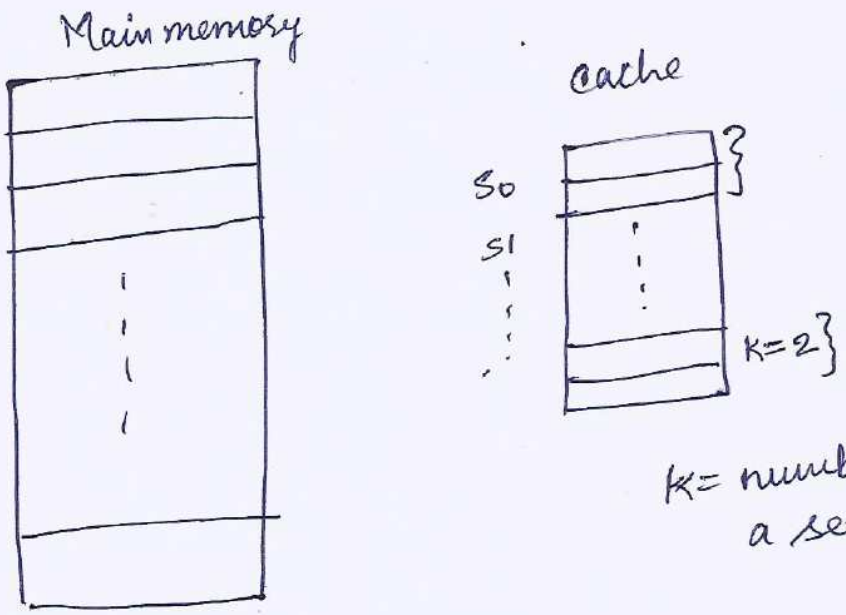
In set associative mapping, ~~the~~ cache lines are grouped in sets. Grouping is done in K-way where K is the number of lines in a set.

> Set associative mapping technique is the combination of direct mapping and associative mapping.

In this mapping the memory block are connected to a set using the logic of direct mapping ~~on~~ (this mapping is from main memory to set). and inside the set, the mapping is associative mapping.

Advantage: The advantage is, the associative mapping is only for a set. The number of lines in a set is only k. Therefore the number of comparisons will be less. So the search time will be reduced.

So the set associative is better than full associative mapping.

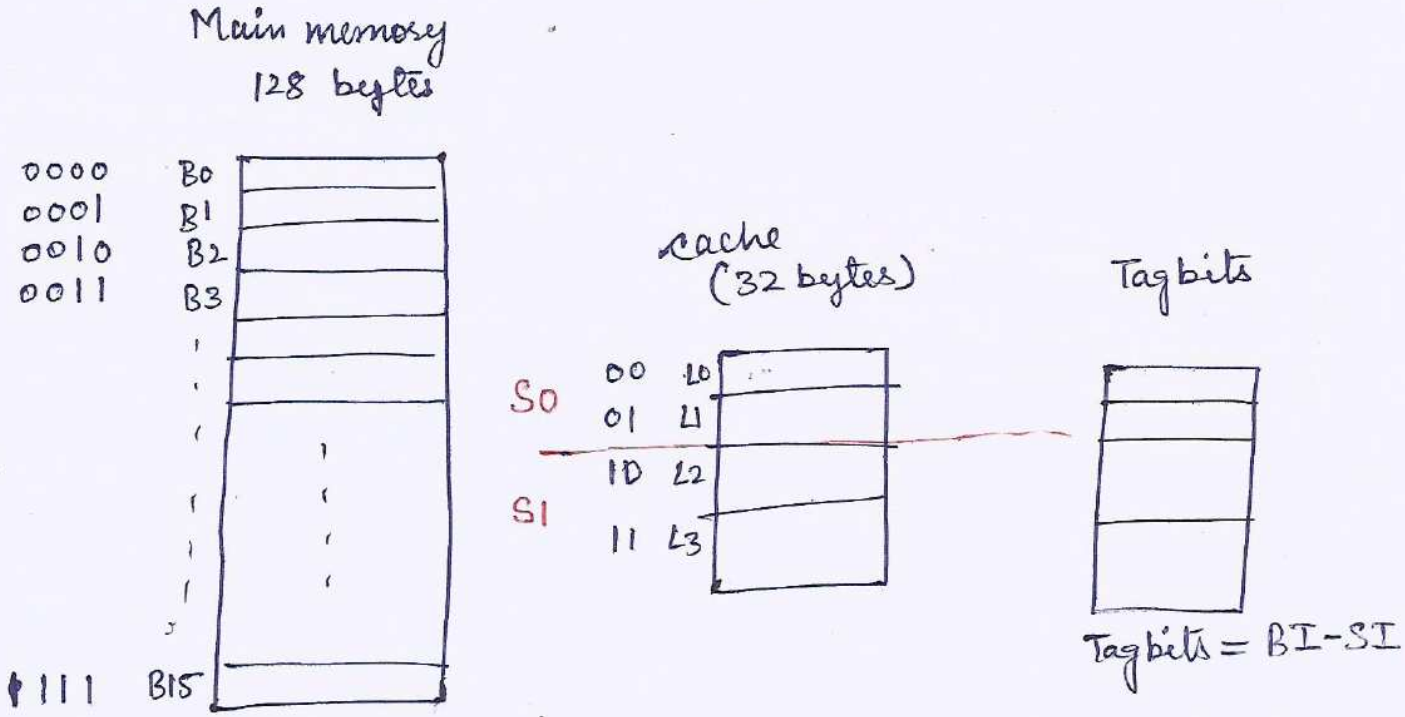


Grouping
K-way
~~Hasarkas~~
e.g ⇒ 2-way set

K= number of lines in a set.

Example

Block size = 8 bytes.



$$\text{Number of block} = \frac{\text{Main memory size}}{\text{Block size}} = \frac{128 \text{ bytes}}{8 \text{ bytes}} \Rightarrow 16$$

$$\text{Number of lines} = \frac{\text{Cache size}}{\text{Block size}} = \frac{32 \text{ bytes}}{8 \text{ bytes}} = 4$$

2-way set associative mapping $\Rightarrow K = 2$ (number of lines in a set = 2)

$$\text{Number of sets} = \frac{\text{Number of lines}}{K} \Rightarrow \frac{4}{2} = 2$$

Number of bits to identify sets = $\log_2$ (Number of sets)

$$(SI) = \log_2(2)$$

$$= 1 \text{ bit}$$

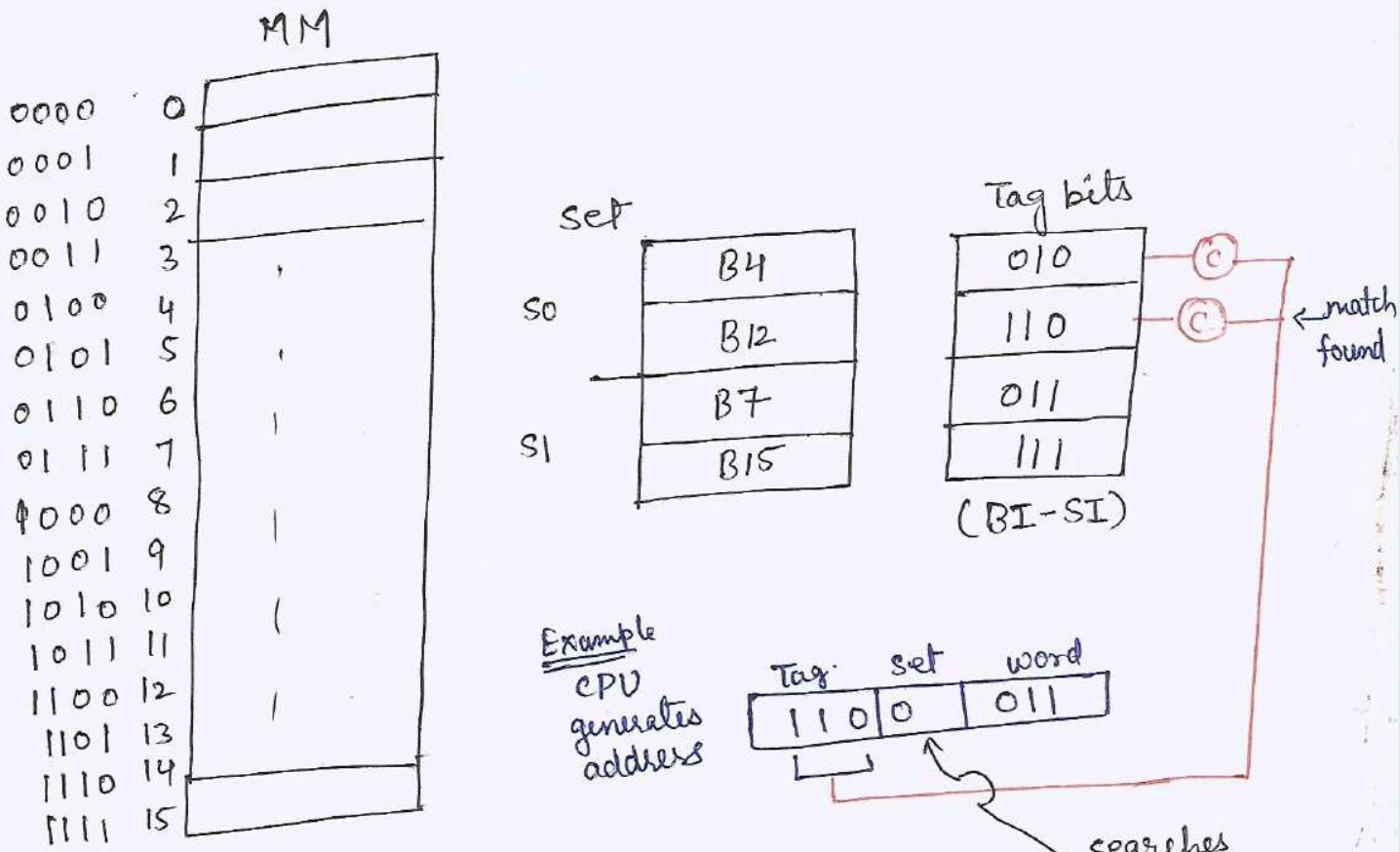
0 | 0

Block can be placed in sets according to

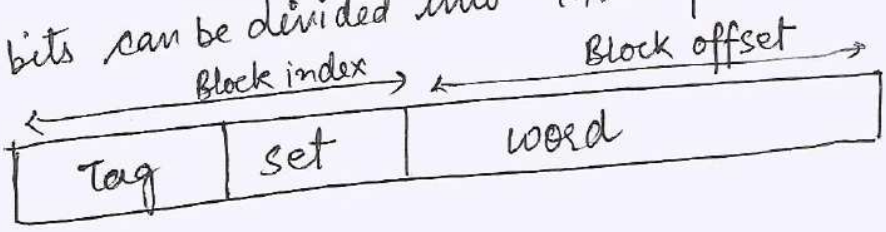
$$S_i = B_j \bmod (\text{number of sets})$$

For this example

Set	Block placed
S0	B0, B2, B4, B6, B8, B10, B12, B14.
S1	B1, B3, B5, B7, B9, B11, B13, B15



Address bits can be divided into Three parts



0 0

Write Policies:

① Write Through:

In this policy, whenever processor writes data into cache memory location, both cache and underlying memory location are updated.

→ Memory is consistent with cache.

→ forces all the writes to update cache and main memory.

② Write Back:

→ In this policy, ~~when~~ data is written only into the cache, not in memory.

→ The main memory is not updated until the cache block needs to be replaced.

⇒ Each block in a write back cache needs a $\boxed{\text{dirty bit}}$ to indicate whether or not it must be saved to main memory before being replaced.

⇒ Better performance

⇒ cache and main memory becomes inconsistent.

Dirty Bit:

cache controller keeps track of which locations in the cache has been changed by putting an extra bit per cache line called "dirty bit."

0 | 0

Virtual Memory.

→ Virtual memory is a technique that lets the execution of different processes which are not totally in main memory

→ One main benifit of this method is that programs can be larger than the physical memory (main memory).

→ The virtual address space of any process is defined as the logical (or virtual) view of how any process gets stored in memory. Normally this view is where a process begins at a certain logical address (say address location 0) and then exists in continuous memory.
Although, the fact is physical memory (main memory might be structured in the form of page frames.
and physical page frames are assigned to a process may not be adjacent.

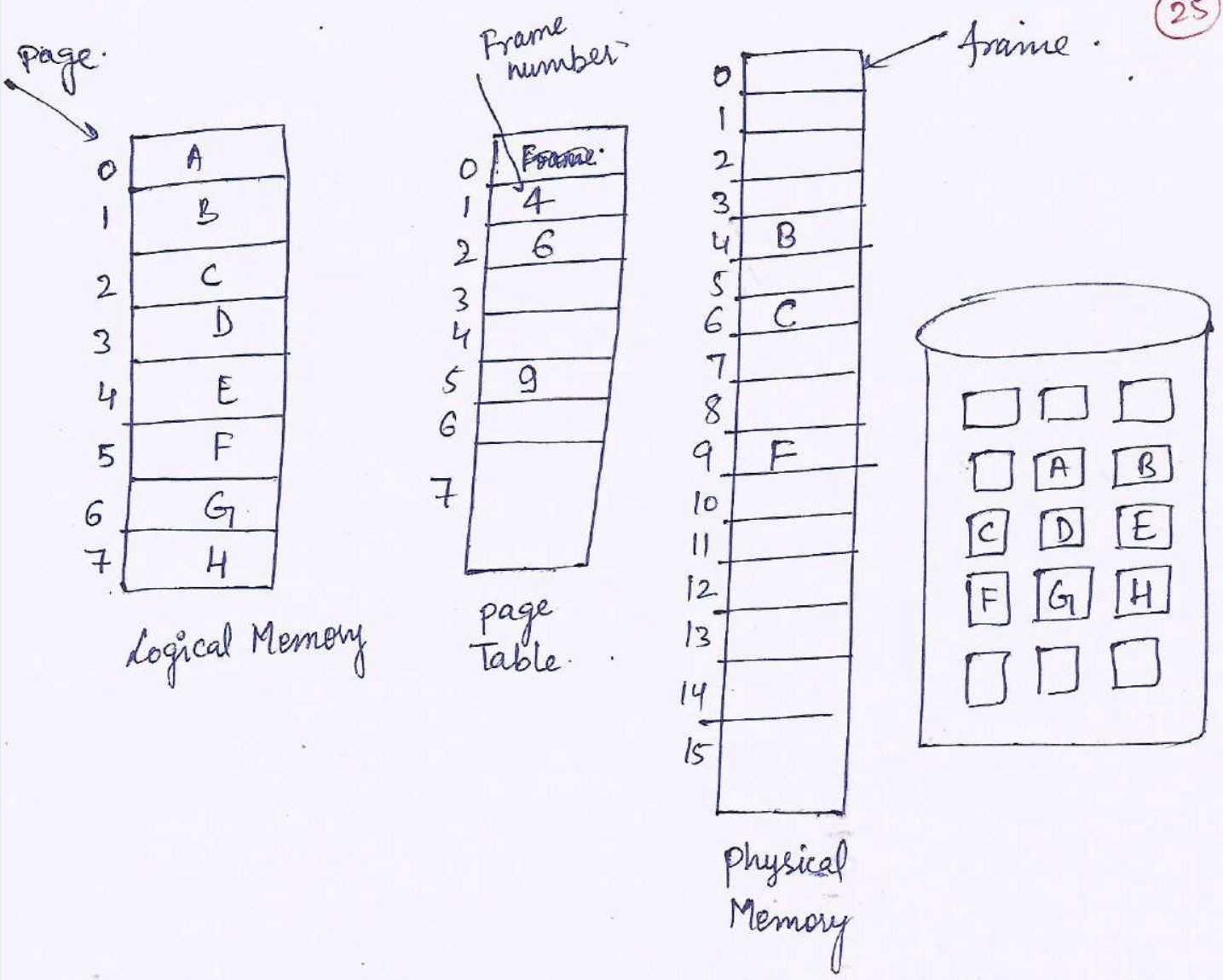
→ Memory Management Unit (MMU) maps the logical pages to physical page frames in memory.

page = logical memory divided into fixed sized block called pages.

frame ⇒ physical memory divided into fixed sized blocks called frames

page size = frame size.

0 | 0



when a page is referenced. if it is found in main memory, then page hit occurs. If not then a page miss occurs, and is called a page fault.

⇒ A computer can address more memory than the amount physically installed on the system. This extra memory actually called a virtual memory and it is the section of a hard disk that's set up to emulate the computers RAM.

0 | 0

Replacement Algorithm:

→ when a page fault (page being accessed is not in memory), the operating system has to choose a page to remove from memory to make space for the page that has to be brought in.

→ The criteria for selecting a page for replacement constitute a replacement policy.

→ The main goal is choosing a replacement policy to maximize the hit ratio.

→ Replacement Algorithms are:

→ FIFO

→ LRU

→ optimal

First In First Out Page Replacement Algorithm:

- FIFO select a page for replacement which has been least recently loaded.
- Easy to implement

→ It does not give high hit ratio

Ques Assume that memory consists of 3 frames and following pages are referenced

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

Show the behaviour of page replacement using FIFO.

0 | 0

Solution Number of frames = 3 . initially all frames are empty

Reference string

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

F denotes page fault occurred. page 0 ~~was~~ was not available in frames

H denotes page Hit as page 3 found in frames.

1	7	0	1
	2	2	1
	1	0	0
	7	7	7
H	F	F	F

Total number of page fault = sum of all page faults.

= 15

0 | 0

Least Recently Used :

→ LRU policy replaces the page in memory that has not been referenced for longer time.

→ This algo replaces a page that has been unused for the longest time

Reference string Given :

{Number of frames = 3 , all frames initially empty}

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

Here 2 replaces 7 because Most recently used 1, then page 0 and least recently used is page 7

Number of page fault = 12

Optimal Page Replacement Algorithm :

- In optimal page replacement policy, that page is replaced which will not be used for the longest period of time.
- It has lowest page fault.

→ This technique is not implemented because it looks into future references.

0 | 0

Number of frames = 3

Reference string:

At this point, page 7 will be replaced

So page 3 will be replaced

	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	0	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
	F	F	F	F	H	F	H	F	H	H	F	H	H	F	H	H	H	F	H	H

Number of page fault = 9

Ques 2: Consider a reference string 1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3. The number of frames in memory are 4. Find out the number of page faults respective to: optimal page replacement, FIFO and LRU page replacement algorithm.

Solution: Number of frames = 4

FIFO Algo.

Reference string:

	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
	1	1	1	1	1	1	1	1	6	6	6	6	6	1	1	1	1	1	1	1
		2	2	2	2	2	2	2	2	2	2	2	2	6	6	6	6	6	6	3
			3	3	3	3	3	3	3	3	5	5	5	3	3	3	3	2	2	2
						5	5	5	5	5	5	6	3	5	5	6	6	5	4	4
	F	F	F	H	H	F	H	H	F	H	H	H	H	F	H	H	H	F	F	F

Number of page fault = 9

0 | 0

LRU:-

Number of frames = 4

Reference string:

page 1, 2, 5 will not be replaced

	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
	1	1	1			5			5				5	5				2	2	2
		2	2			3			6				6	6				6	6	3
			3			2			2				2	1				1	1	1
						1			1				3	3				3	4	4
	F	F	F	H	H	F	H	H	F	H	H	H	F	F	H	H	H	F	F	F

at this point page 6, 5, 2 will not be replaced

Number of page fault = 10

Optimal Page Replacement

Number of frames = 4

Reference string:

at this point page 1 will be replaced

	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
	1	1	1			5			5					1					1	
		2	2			3			3					3					3	
			3			2			2					2					4	
						1			6					6					6	
	F	F	F	H	H	F	H	H	F	H	H	H	H	F	H	H	H	H	F	H

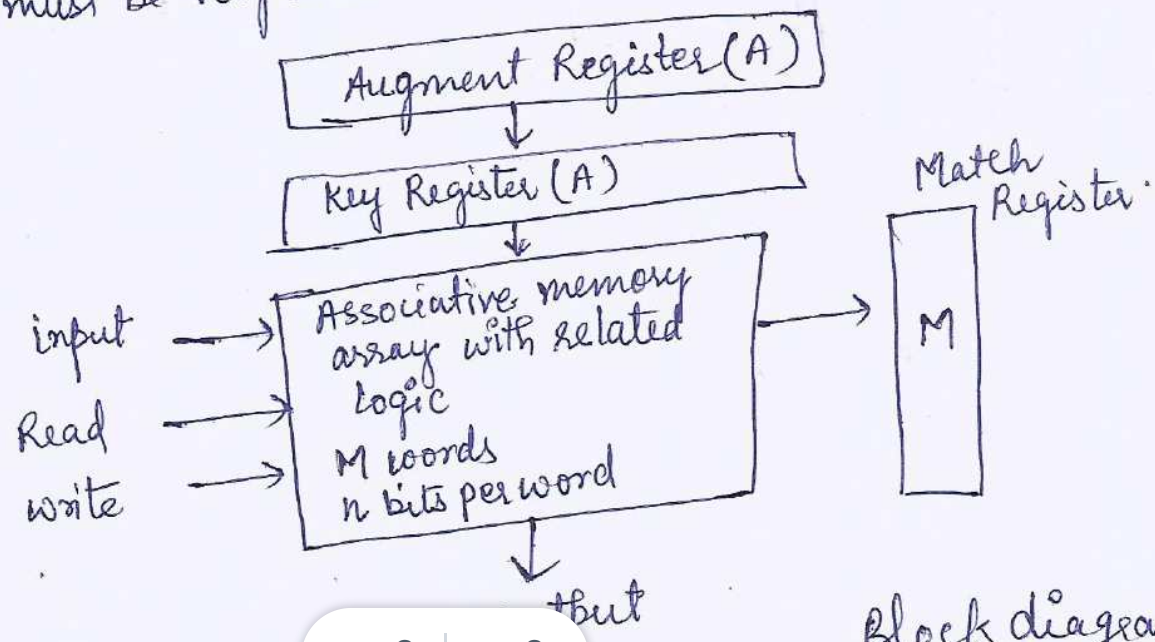
you can replace any page except page 3 or may use FIFO concept

Number of page fault = 7

→ What is associative memory?

Associative Memory:

- Associative memory is a type of memory from which items may be retrieved by matching some part of their content rather than by specifying their address. hence also called associative storage or Content Addressable Memory (CAM).
- This type of memory is accessed simultaneously and in parallel on the basis of data content rather than any specific address or location.
- When a word is to be read from an associative memory the content of the word or part of the part is specified.
- The memory locates all the words which match the specified content and marks them for reading
- Because of its organization the associative memory uniquely is suited to do parallel searches by data association.
- It is more expensive than RAM.
- used in application where the search time is critical and must be very short.



Block diagram

0 | 0

Auxiliary Memory

→ Auxiliary memory are external memory which can store data permanently

→ used for backup purpose.

→ It is economical and larger in size.

Types of auxiliary Memory:

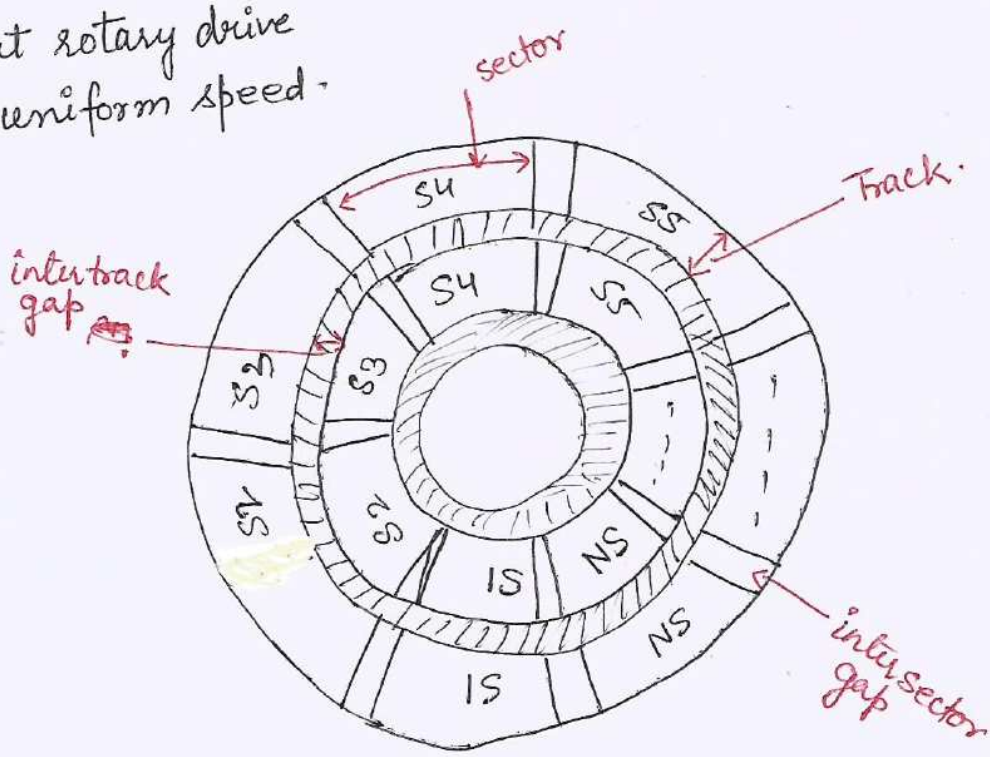
- Magnetic Disk
- Magnetic Tapes
- Optical Disks

Magnetic Disk:

physical characteristics: Magnetic disk is circular made of plastic and coated with magnetized material.

→ it is mounted at rotary drive

→ rotates at uniform speed.



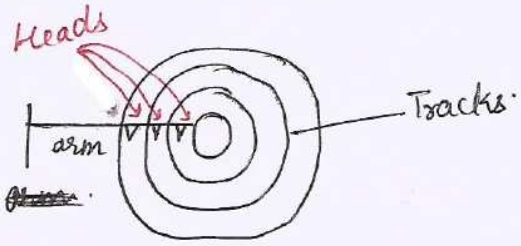
Disk data layout

0 | 0

Fixed Head vs Movable Head Disk:

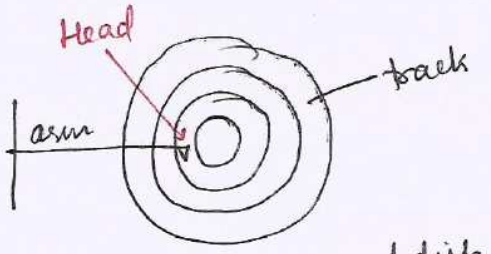
→ In a fixed head disk, there is one read-write head per track.

→ All the heads are mounted on a rigid arm that extends all the tracks.



Fixed head disk

In a movable head disk, there is only one read-write head. Head is mounted on arm. Head must be able to be positioned above any track, the arm can be extended or retracted for this purpose.

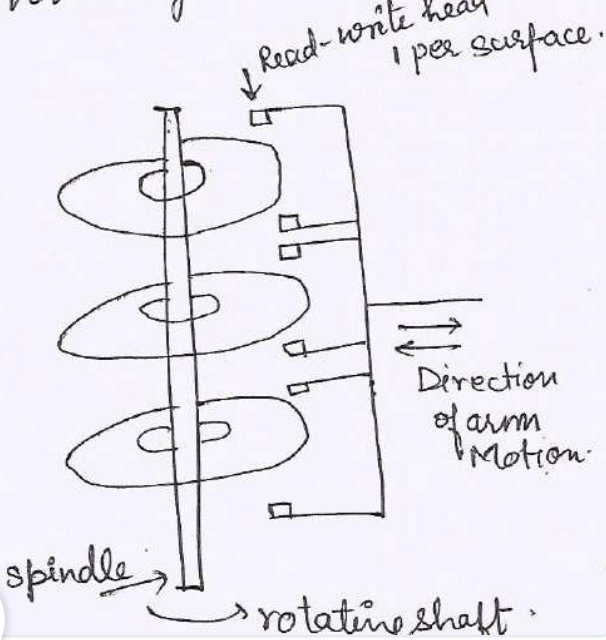


movable head disk

Double sided disk

→ The magnetized coating is applied to both the sides of the ~~platter~~ plates.

⇒ **Disk pack:** multiple platers stacked vertically.



0 | 0

Disk Performance Parameters

When the disk drive is operating, the disk is rotating at a constant speed. To read or write, the head must be positioned at the desired track and at the beginning of the desired sector on that track.

* Track selection involves moving the head in a movable head system or electronically selecting one head on a fixed-head system.

→ On a movable-head system, the time it takes to position the head at the track is known as **seek time**.

→ In either case the track is selected, the disk controller waits until the appropriate sector rotates to line up with the head.

→ The time it takes for the beginning of the sector to reach the the head is known as **rotational delay or rotational latency**.

→ The sum of seek time, if any and the rotational delay equals **access time**, which is the time ~~to position~~ it takes to get into to read or write.

→ Once the head is in position, the read or write operation is then performed as sector moves under the head; this is the data transfer portion of the operation, the time required for the transfer is the **transfer time**

0 | 0

Seek Time:

→ time required to move the disk arm to the required track.

Rotational Delay

→ time required to reach the beginning of the sector.

$$\boxed{\text{Rotational latency} = \frac{1}{2r}}$$

where r = rotation speed, (in revolution per second)

Transfer Time:

→ Transfer time to or from the disk depends on the rotation speed of the disk in the following fashion.

$$\boxed{T = \frac{b}{rN}}$$

where

T = Transfer time

b = number of bytes to be transferred

N = Number of bytes on a track.

r = rotation speed, in revolutions per second.

Total Average access time

$$\boxed{\text{Total Average Access time} = \text{Seek Time} + \text{Rotational latency} + \text{Transfer Time}}$$

0 | 0

Magnetic Tapes:

→ used to provide backup storage
→ Permanent type of memory
→ External Memory
→ It consists of tape constructed from plastic and is covered with magnetic oxide layer.
→ A tape is sequential access device. If the tape head is positioned on ~~recod~~ record 1, then in order to read number N, it is necessary to read ~~recod~~ records from record 1 through (N-1) to reach record N.
→ Information stored on tape is organized into blocks. And blocks are separated through with record gap.

Optical Disks:

→ optical memories have a huge capacity. Here, information is written using laser beam.
→ These memories stores huge amount data and are very reliable.

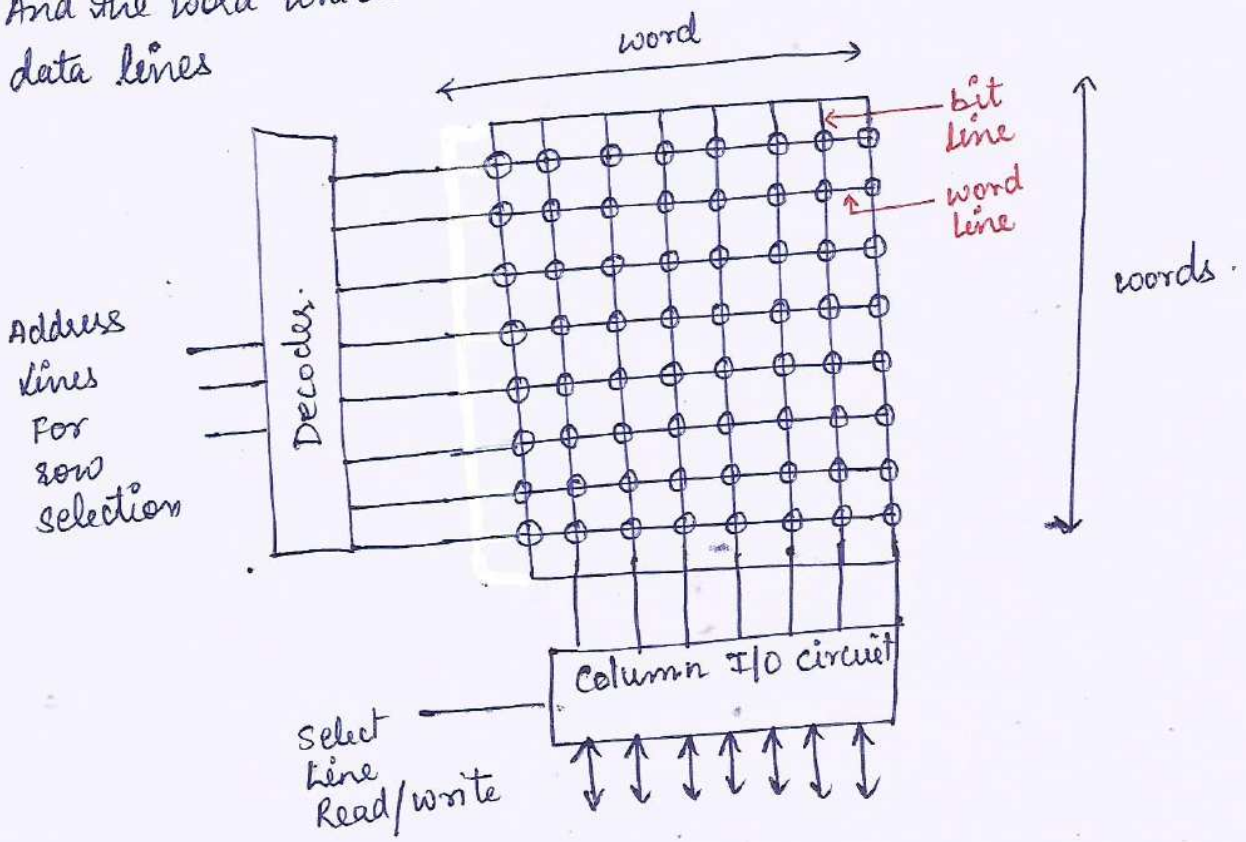
CDROM (compact Disk ROM)

→ very reliable
→ Error Correction facility.
→ A huge intensity laser beam is used.
→ Information storage is done evenly across the disk in segments of the same size. This is achieved by rotating disk at variable speed. This is called constant Linear Velocity.

0 | 0

2D Memory Organization :

In 2D organization, memory is divided in the form of rows and columns. Each row contains a word. A decoder is used to select the row which address is contained in MAR (Memory Address Register).
And the word which is selected will be read or written through the data lines



memory size in this example ⇒ 8×8 → each word size.
↓
8 number of words

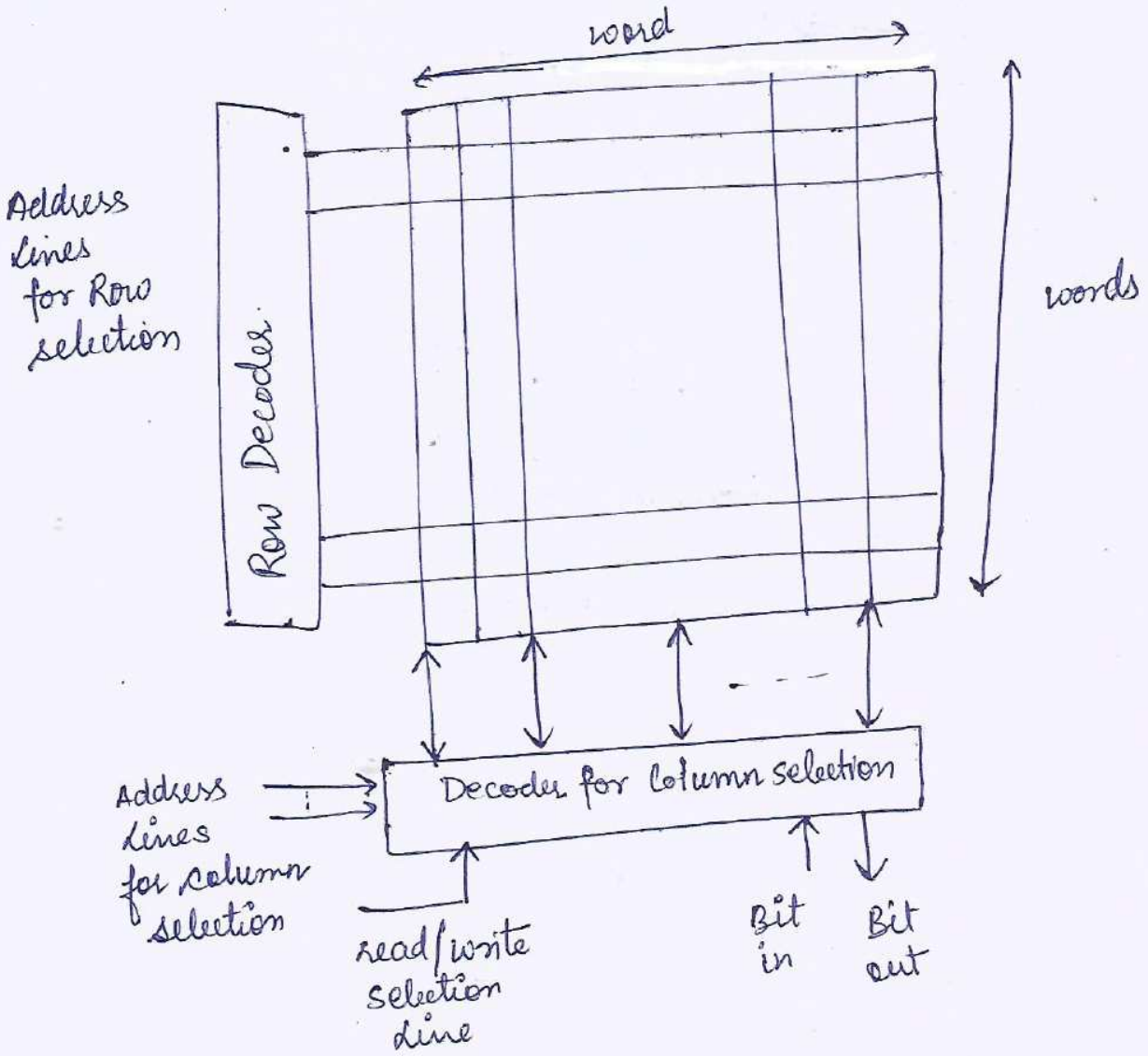
2D Memory organization

$2\frac{1}{2}$ D Memory Organization:

→ This organization, two decoders are used.

→ One decoder to select the row (or words) and the column decoder selects the column.

→ The address from MAR will go in row decoder inputs. The decoders will select the respective cell. Through the bit outline, the data from that location will be read or through bit in line data will be written at that memory location.



0 | 0