bandwidth : is the Data transfer capacity of a computer Network in bits per second (Bps).

Unit - III

Memory Interleaving : (memory Modules)

⇒ It is a Technique that divides memory into a number of modules such that successive words in the address space are placed in the Different Modules.

⇒ Memory interleaving increases bandwidth by allowing simultaneous access to more than one chunk of memory.

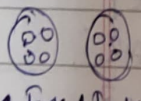
⇒ This improves performance because the processor can transfer more information to/from memory in the same amount of time.

word = 2byte

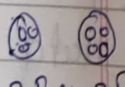
Divides system memory into multiple Blocks.

2 Methods → 2 way Interleaving
(2 memory Blocks are accessed at same time for Reading & writing ope)

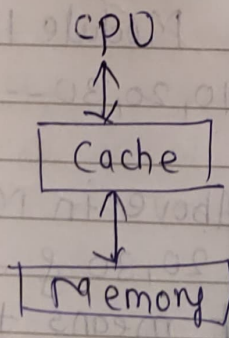
4 way Interleaving
(4 memory Blocks are accessed at same time)

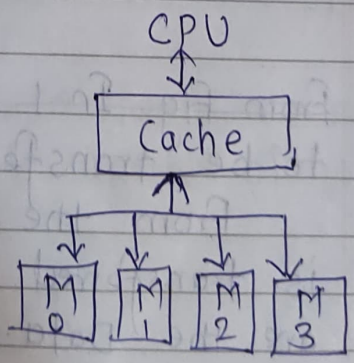


= 2way
1 people



= 4way
2 people





How it is done ?

By spreading memory addresses evenly across memory Modules.

High order bits | Low order bits

Fig. Consecutive Word in a Modu

0	0000	10
1	0001	20
2	0010	30
3	0011	40
4	0100	50
5	0101	60
6	0110	70
7	0111	80
8	1000	90
9	1001	100
10	1010	110
11	1011	120
12	1100	130
13	1101	140
14	1110	150
15	1111	160

MM

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

2^4

e.g. 4-way interleaved mem

↓

4 Modules

Let's Assume 16 Data's to be Transfered to the 4 Module where Module 00 be Module 1
Module 01 be Module 2
Module 10 be Module 3
Module 11 be Module 4.

~~From Fig. In 1~~ Also 10, 20, 30 --- 160 are the Data to be transfered.

From the fig. Above in Module 1, 10 [data] is transfered then 20, 30 & finally 40 which are the Data. That means the data are added consecutively in the Module till its max capacity

Most significant bit (MSB) provides the Address of the Module

Least significant bit (LSB) provides the Address of the Data (in the Module)

MSB	LSB
10	00

27/9/22 → 13,

Module Data
MSB LSB

For Example, to get go(Data) 1000 will be provided by the processor.

Module: 1 contains Data : 10, 20, 30, 40
" 2 " " : 50, 60, 70, 80
" 3 " " : 90, 100, 110, 120
" 4 " " : 130, 140, 150, 160

Consecutive Word in Consecutive Module

	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

Least Significant Bit (LSB) : Address of Module
(MSB) : Address of Data

MSB	LSB
10	00

* A memory is a storage unit - It is a place to hold data & instructions.

Memory Location: Memory is divided into multiple small parts of fixed size & capable of holding info

* Hierarchy of Memory

A Memory unit is the collection of storage units or devices together. The memory unit stores the Binary information in the form of bits. Generally Memory / storage is classified into 2 categories.

RAM ⇒ Volatile Memory: This loses it data, when power is switched off.

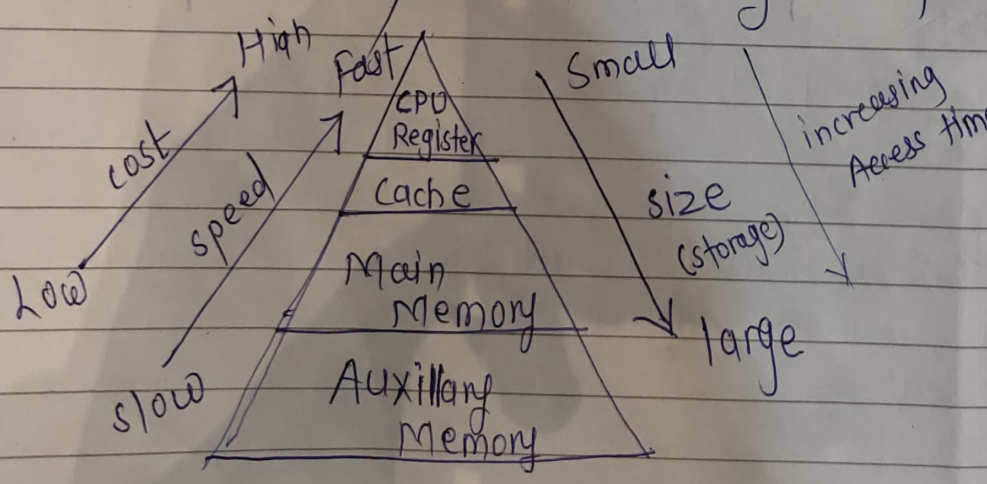
ROM ⇒ Non Volatile Memory: This is a permanent storage & does not lose any Data when power is switched off.

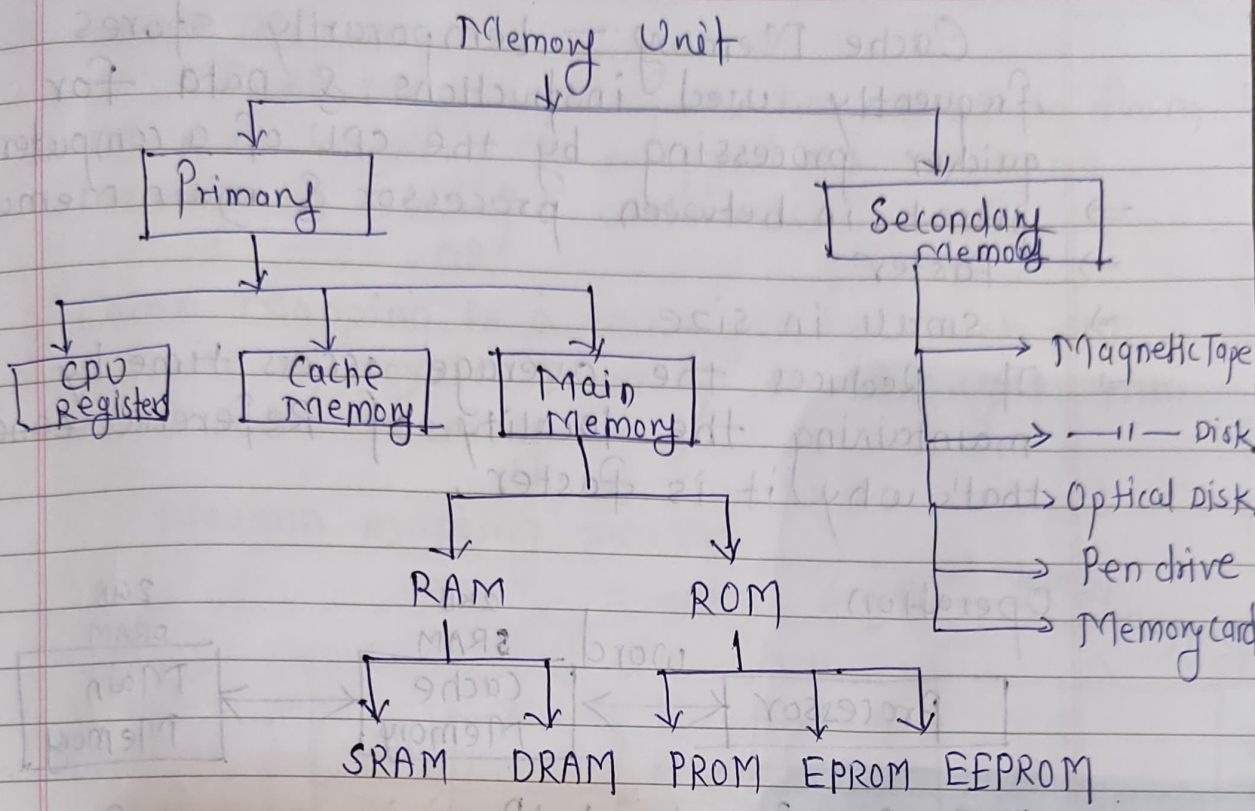
⇒ The total Memory capacity of a Computer can be visualized by hierarchy of components.

⇒ The Memory hierarchy system consists of all storage devices contained in a computer system.

External Memory or Secondary Memory
— Magnetic Disk, Optical Disk, Magnetic Tape
i.e. peripheral storage devices which are accessible by the processor via I/O Module.

Internal Memory or Primary Memory -
Main Memory, cache Memory & CPU Registers.
— This is Directly Accessible by the processor.





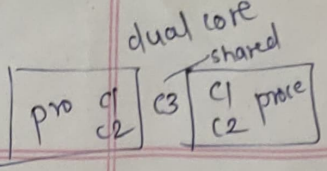
Secondary (Auxillary) Memory

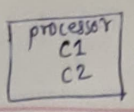
- Magnetic Tape (Seq. Access)
- Magnetic Disk → HDD, FDD
- Optical Disk → CD, DVD, BRD, HVD
- Flash Memory → Memory card, Pendrive

PROM Programmable Read only Memory
EPROM Erasable
EEPROM Electrically erasable

Pocket Money | Cupboard | Bank

Cache mem | RAM | Hard disk

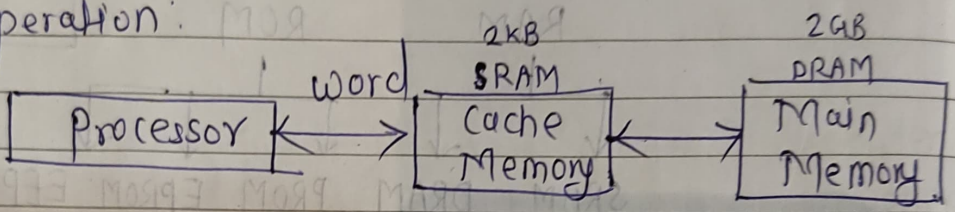




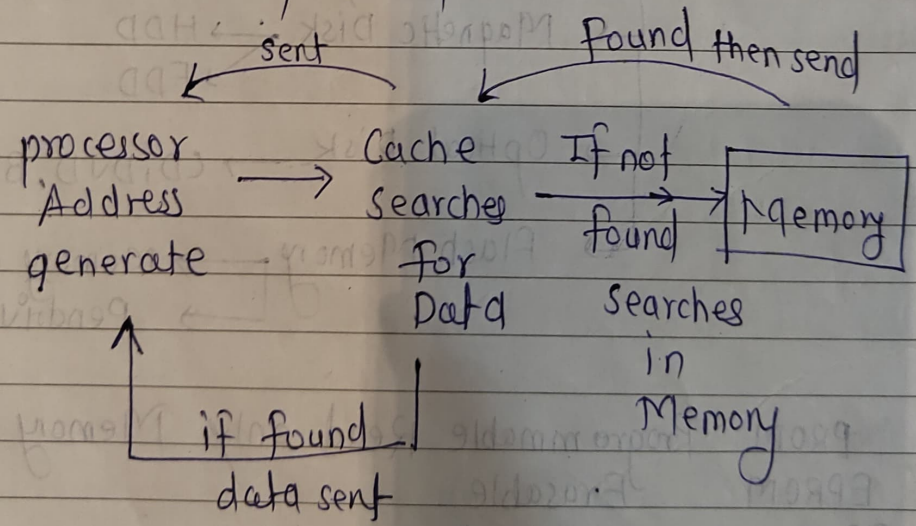
Cache Memory :- temporarily stores frequently used instructions & data for quicker processing by the CPU of a computer.

⇒ present in between processor & main memory
⇒ faster
⇒ small in size
⇒ It Reduces the average access time by maintaining the locality of Reference ~~concept~~ that's why it is faster.

Operation:



⇒ Transfer of Data bet^n main memory & cache memory is Block by block
⇒ Transfer of Data bet^n cache & processor is word by word.



cache hit → Address/data is found in cache memory
cache miss → ——//—— is not found.

The process of transfer the Data from main memory to cache Memory is called as mapping

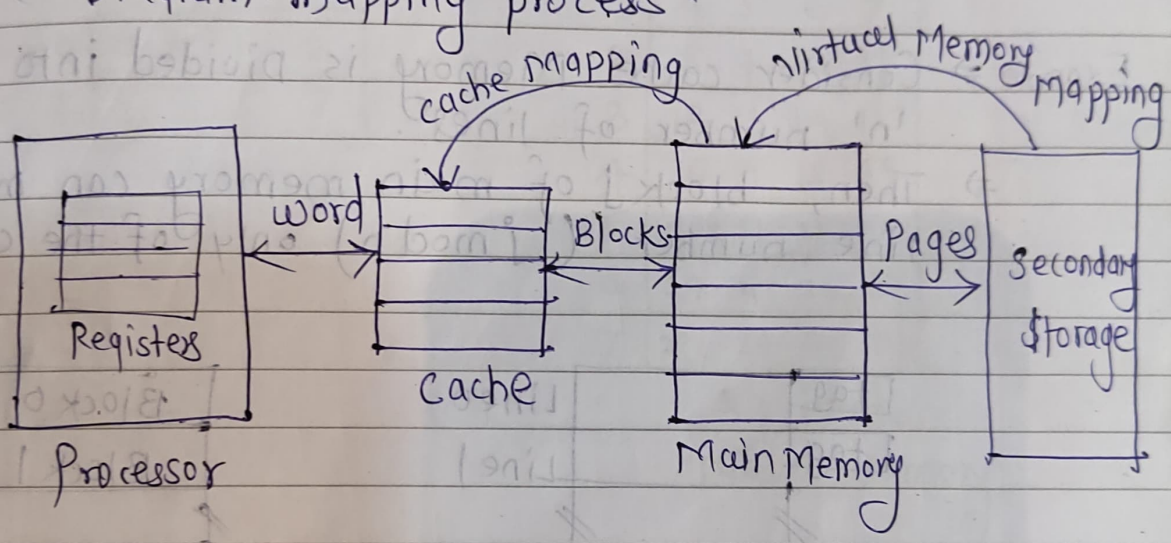
Cache Mapping

Cache Mapping defines how a block from the main memory is mapped to the cache memory in case of a cache miss.

OR

Cache Mapping is a technique by which the contents of main Memory are brought into the cache Memory.

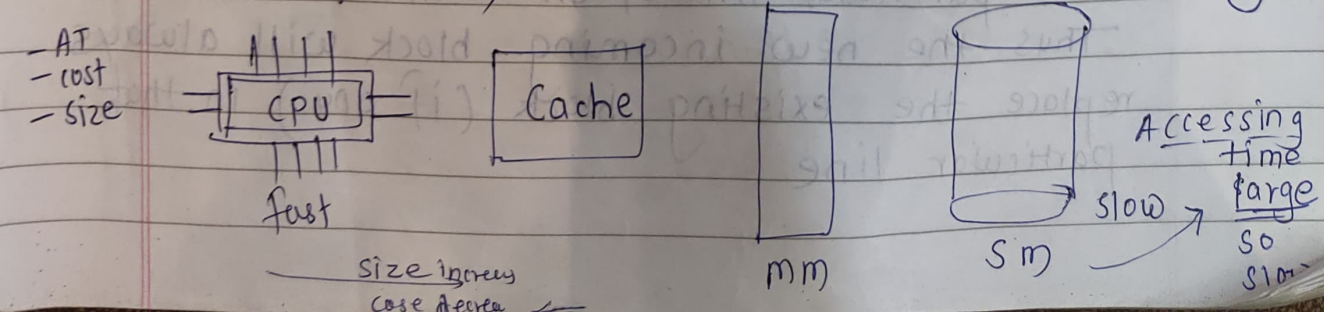
Diagram. mapping process.



⇒ Main Memory is divided into equal size partitions called as blocks or frames

⇒ Cache Memory is divided into partitions having same size as that of blocks called as lines

⇒ During cache Mapping, block of main Memory is simply copied to the cache & the block is not Actually brought from main Memory



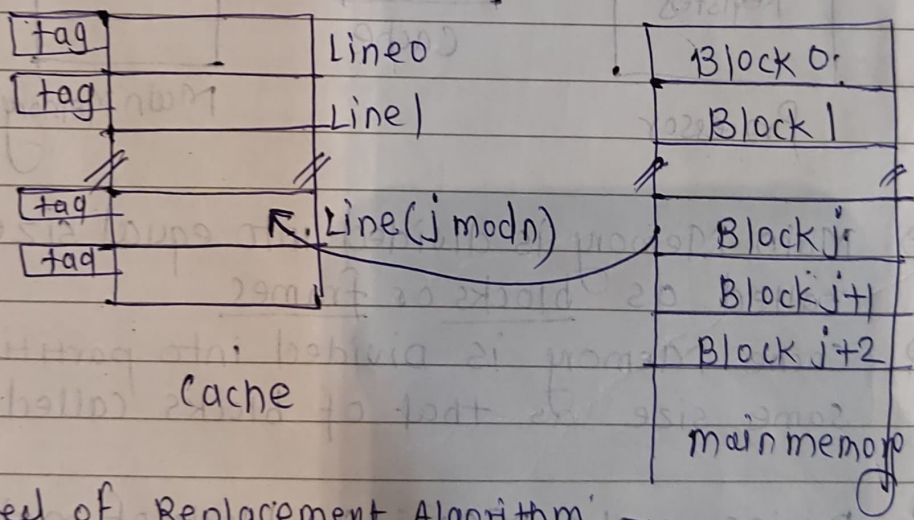
Cache Mapping Tech.

Direct | fully Associative | K-way set Associative

1) Direct Mapping : In Direct Mapping
⇒ A particular block of main Memory can map only 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)

Example ⇒ 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

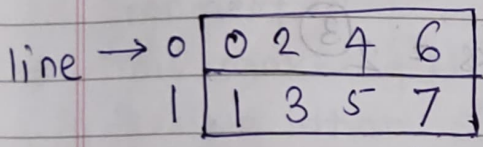


Need of Replacement Algorithm :-
There is no need of any Replacement Algorithm because a main memory block can map only to a particular line of the cache.
Thus the new incoming block will always replace the existing block (if any) in that particular line

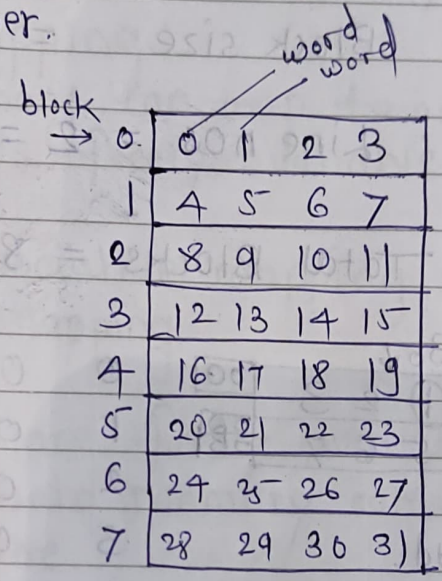
In Direct Mapping, the physical Address is divided as

Tag	Line number	Block/Line offset

Block Number.



8 words.



8 Blocks

32 words

Main memory

1 = word = 1 byte

1 Block size = 4 words

- No. of Blocks = 32/4 = 8 blocks ∴ words/size of block

No of cache line = 8/4 = 2 lines

Cache line size = Block size

Cash line no. = (MM Block No) Mod (No. of lines in cache)

- = (0) mod (2)
  = 0 Remainder → Block 0 ⇒ line 0
- = (1) mod 2 →
  = 1 Remainder
- = (2) mod (2) = 0 (the line)
  (Block)
- = (3) mod (2) = 1

physical Address

Tag	Line no.	Block size
2 bits	1 bit	2 bits

blocks
$2^3 = 8$
Block Number

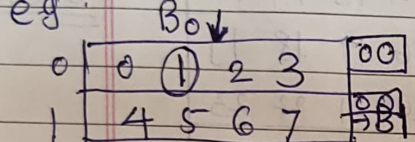
physical Address 32 words $2^{(5)}$ = 5 bits

Block size = 4 words = $2^{(2)}$ = 2 bits

Line nos. = 2 = $2^{(1)}$ = 1 bit

Total Blocks = 8 blocks = $2^{(3)}$

eg.



0	0 0 0 ✓	0 0 0th
1	0 0 1	0 1 1st
2	0 1 0 ✓	1 0 2nd
3	0 1 1	1 1 3
4	1 0 0 ✓	
5	1 0 1	
6	1 1 0 ✓	
7	1 1 1	

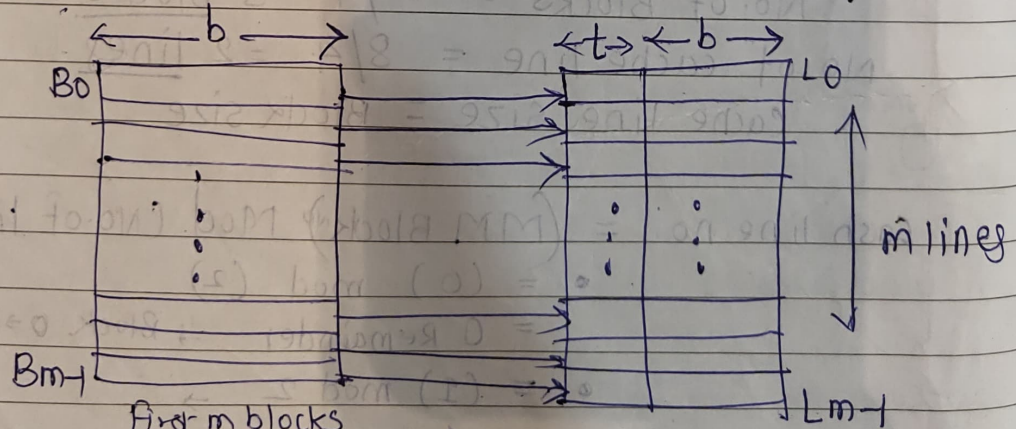
tag | lineNo

Phy Add.

e.g. 00 | 0 | 01 | hit

e.g. 01 | 0 | 01 | miss

Direct Mapping

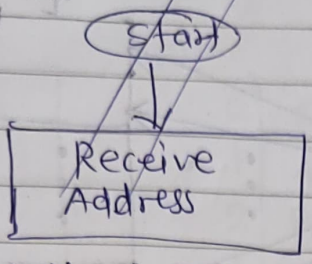


b = length of Block in bits

t = length of tag in bits

physical Address - refers to a memory address or the location of a memory cell in the main memory.

Cache Read operation.



Fully Associative Mapping:

⇒ A block of main memory can map to any line of the cache that is freely available at that moment.

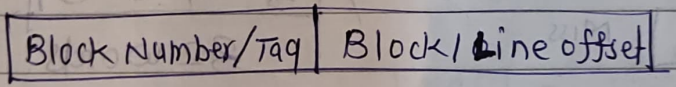
⇒ This makes fully Associative mapping more flexible than Direct Mapping.

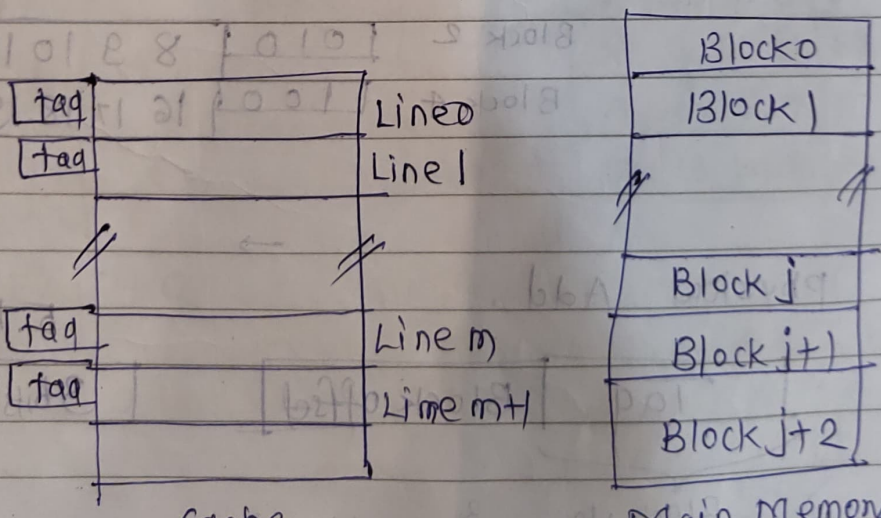
⇒ All the lines of cache are freely available.

⇒ Thus, any block of main memory can map to any line of the cache.

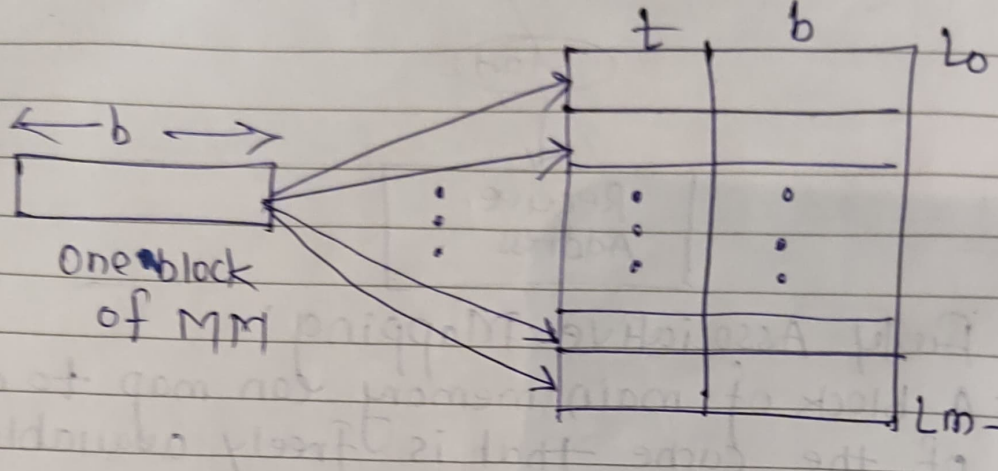
⇒ If all the cache lines occupied, then one of the existing Blocks will have to be replaced.

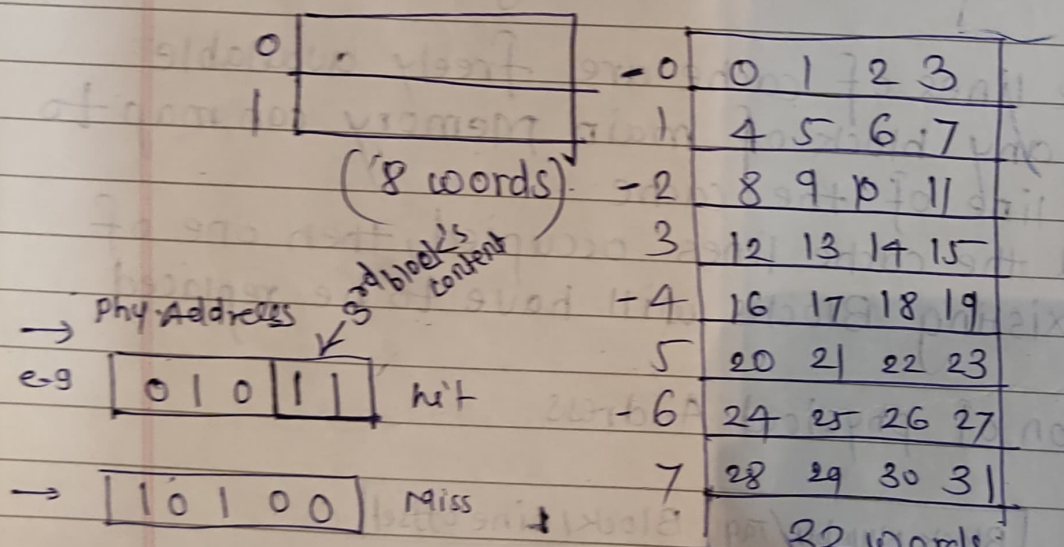
Division of physical Address





Associative Mapping

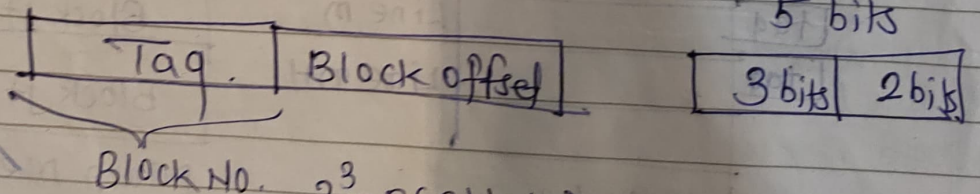




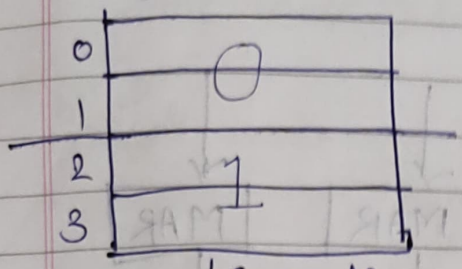
Block 2 | 0 1 0 | 8 9 10 11
Block 4 | 1 0 0 | 16 17 18 19
} initial cache status

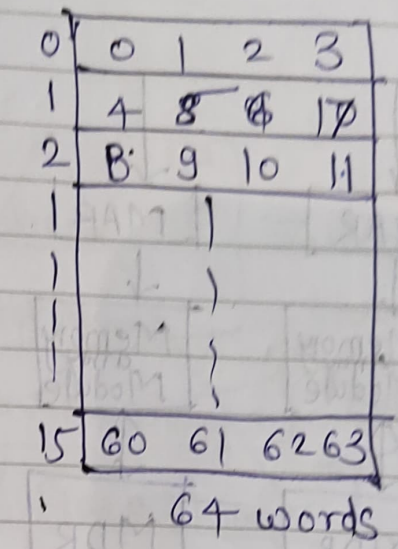
→

physical Add.



Set Associative Mapping





BS = 4 words

NCL = 16

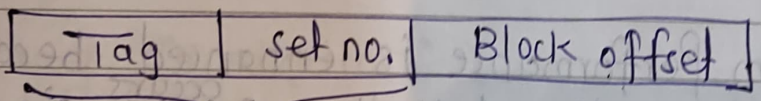
$NBs = \frac{64}{4} = 16$ Blocks

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

$2^6 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 64$

2-way set Associative

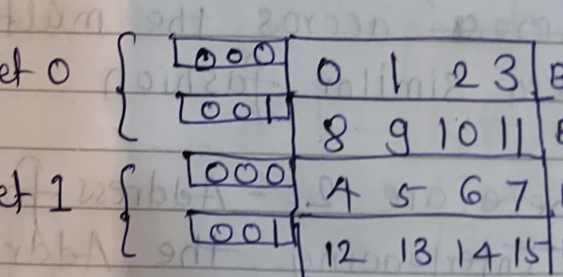
Physical Address

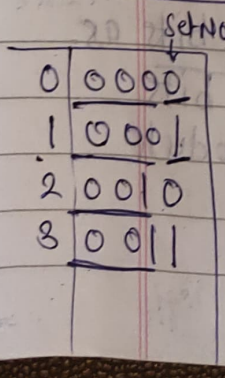


No of sets

set

0 mod 2 → 0 ✓
1 mod 2 → 1
2 mod 2 → 0 ✓
3 mod 2 → 1
4 mod 2 → 0 ✓





⇒ 0 0 1 0 1 0 hit

⇒ 0 1 0 1 0 0 miss

$2^6 = 64 = 6$ bits phy Add

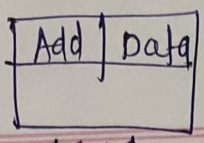
3 | 1 bit | 2 bits

how many sets

$2^1 = 2$

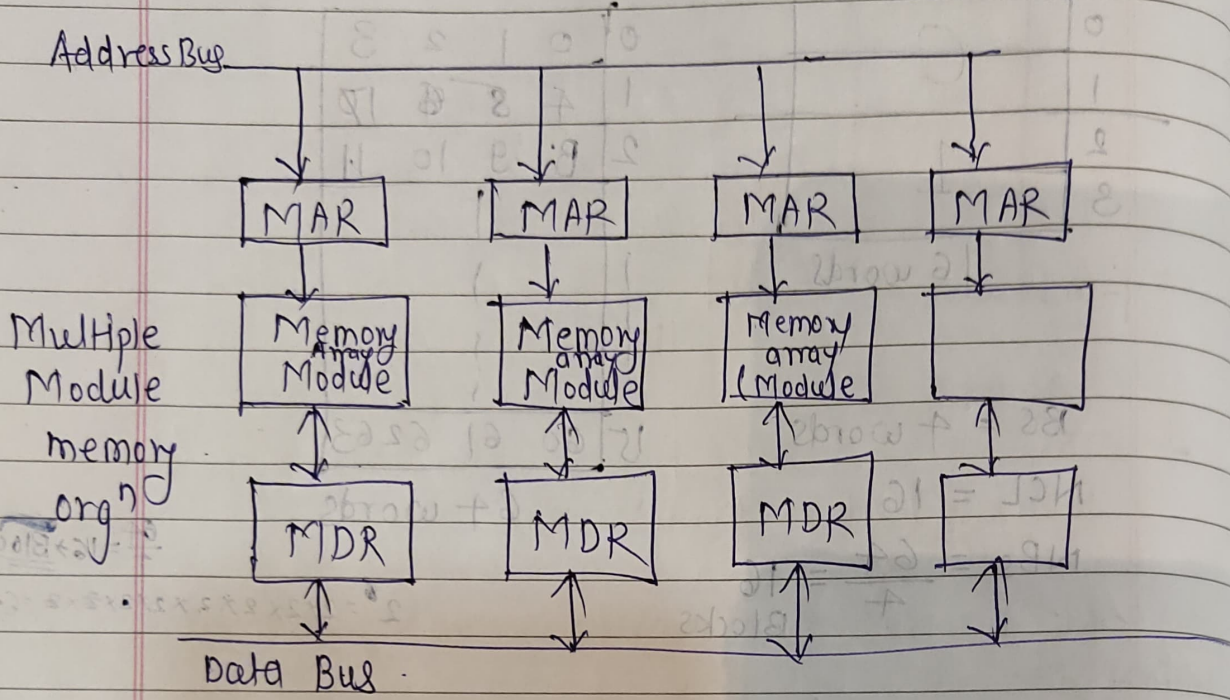
$2^2 = 4$

Shanu Kuttan CSE classes



M.M.

Memory Module is a memory array together with its address & Data registers.

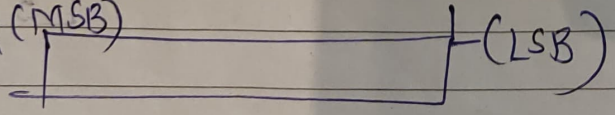


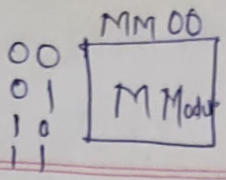
MAR Receives infor^m from a common Address Bus
MDR communicate with a bidirectional data Bus

System Performance is enhanced because read & write Activity ~~Across~~ occurs simultaneously ~~across~~ accros the multiple modules in a similar fashion.

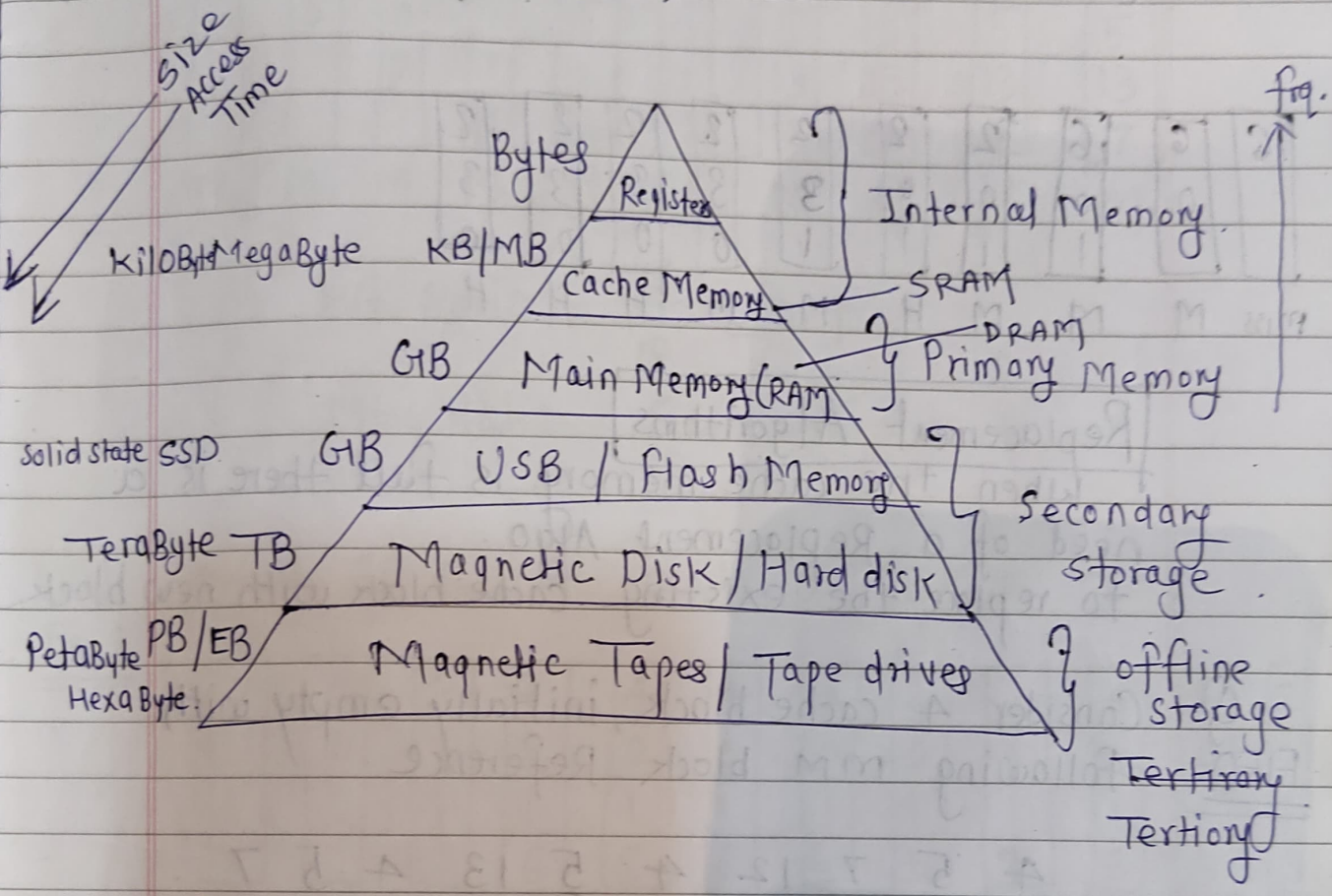
There are 2 - Address formats for Memory Interleaving the Address space.

① High-Order Interleaving : High-order bits as Module Address & the lower order bits as the word address within each module.





Low-order Interleaving: Low-order bits as Module Address & the high-order bits as the word address.

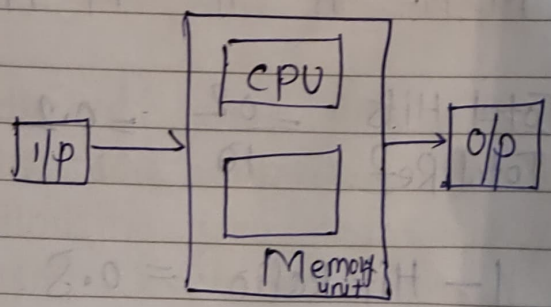


Memory Orgnization:

Diff bet^n Orgnization :- Peripherals Organized.

Architecture

- → K/b, mouse
- → Mouse
- → Memory.



When processor initiates,

① FIFO

② LRU (Least Recently Used)

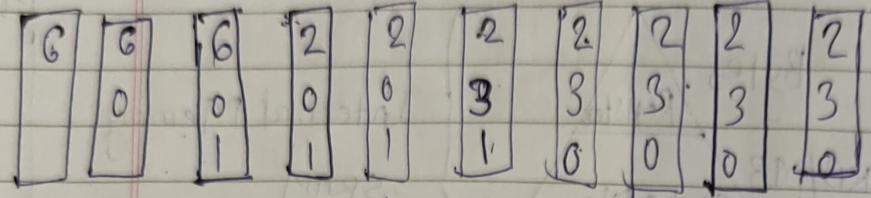
③ Optimal ⇒ study purpose

practically possible

42, 27

Reference string 6 0 1 2 0

6 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 6 0



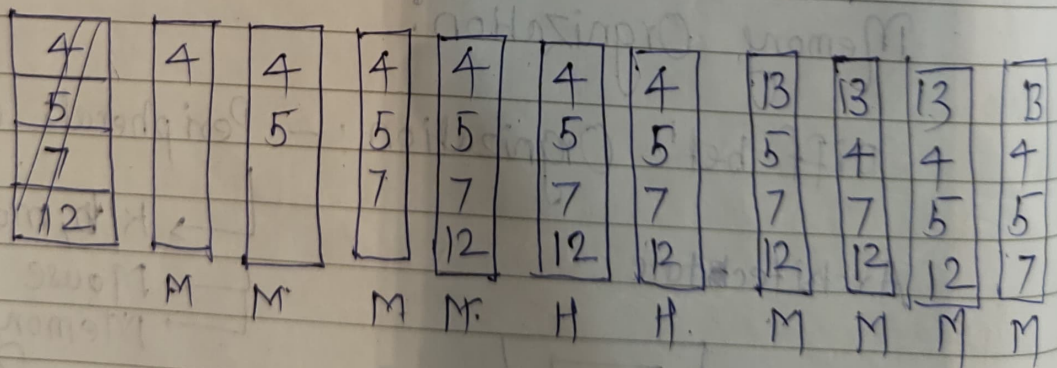
Replacement Algorithms

When the cache memory is full there is a need of a Replacement Algo. to replace the existing cache block with new block

Q. Consider 4 cache block, initially empty with following MM block Reference.

FIFO

4 5 7 12 4 5 13 4 5 7



Hit Ratio = $\frac{\text{Total Hits}}{\text{Total Ref}} = \frac{02}{10} = 0.2$

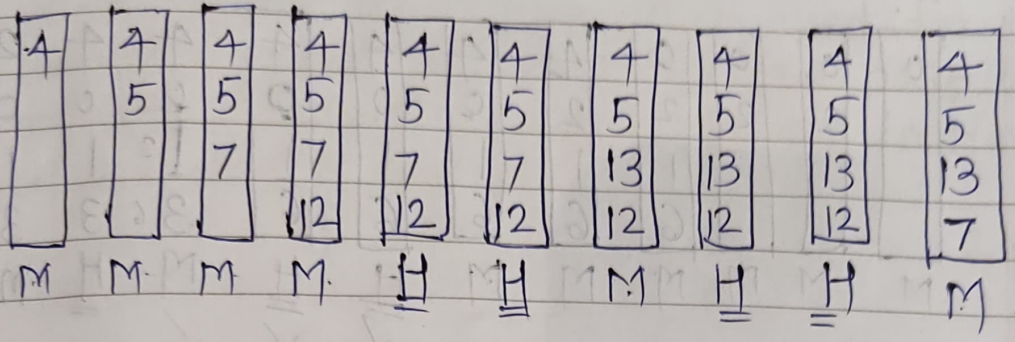
Miss Ratio = 1 − Hit Ratio = 0.8

or = $\frac{8}{10} = 0.8$

page fault = miss

4 5 7 12 4 5 13 4 5 7

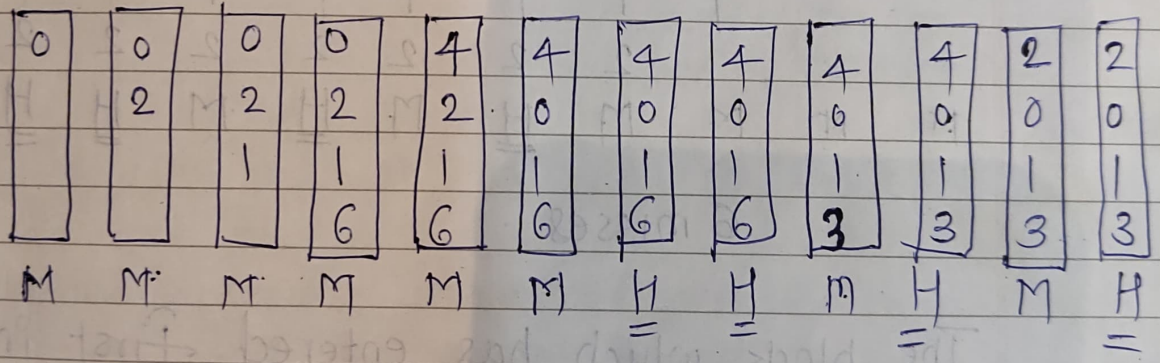
LRU Least Recently Used :



Hit Ratio = $\frac{4}{10} = 0.4$

Miss Ratio = $\frac{6}{10} = 0.6$

LRU Ex. 2 0 2 1 6 4 0 1 0 3 1 2 1



Hit Ratio = $\frac{04}{12} = 0.33$

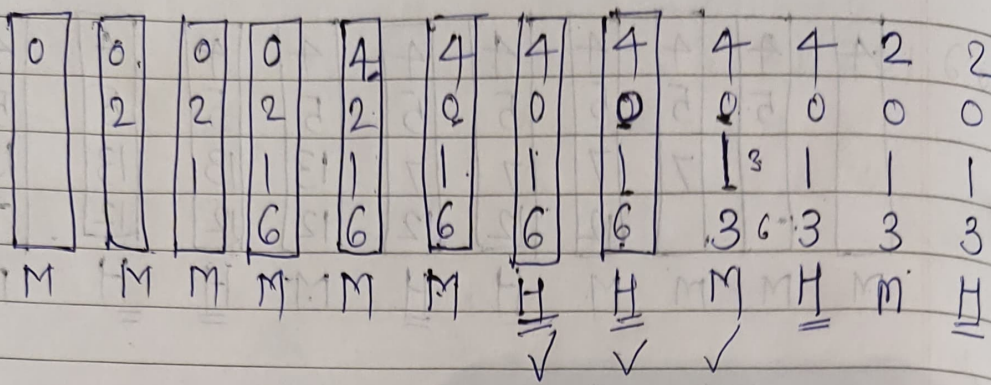
Miss / page fault Ratio = $\frac{08}{12} = 0.67$

discard least Recently accessed content

A LRU cache organizes items in order of use allowing you to quickly identify which item hasn't been used for the longest amount of time

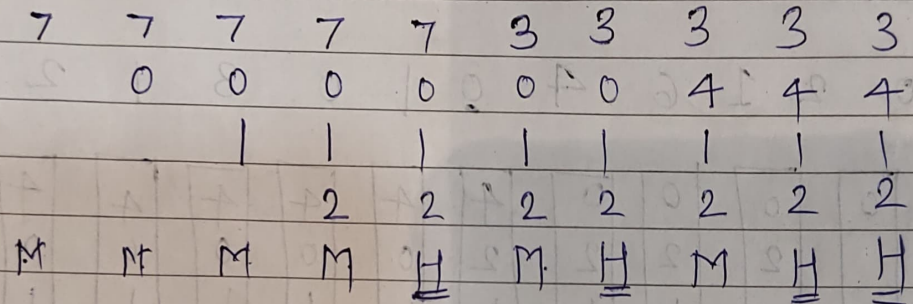
Ex 2 0 2 1 6 4 0 1 0 3 1 2

FIFO



FIFO 7 0 1 2 0 3 0 4 2 3

cache memory has 4 lines.



∴ 6 misses.

The block which has entered first in the main be replaced first.

This can lead to a problem known as "Belady's Anamoly" it starts that if we increase the no. of lines in cache memory the cache miss will increase

LRU (least Recently Used)

The page which was not used for the largest period of time in the past will get reported first. (looking backward in time)

e.g. Let we have a sequence

7 0 1 2 0 3 0 4 2 3

& cache memory has 3 lines

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

← With 4 cache lines

7	7	7	7	7	3	3	3	3	3	
	0	0	0	0	0	0	0	0	0	
		1	1	1	1	1	4	4	4	
			2	2	2	2	2	2	2	6 Misses
M	M	M	M	H	M	H	M	H	H	

Optimal Approach: ~~This~~ The page which will not be used for the largest period of time in future reference will be replaced first.

⇒ This optimal Algorithm will provide the best performance but this is difficult to implement as it Requires the future Knowledge of pages which is not possible.

⇒ Used for comparison studies.

⇒ The use of this algo. guarantees the lowest possible page fault rate for a fixed rate of frames.

FIFO When there is a need for page Replacement, FIFO algo swaps out the page at the front of the queue that is the page which has been in the memory for longest.

Consider the page Reference string of size 12: 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3

with frame size (i.e. maximum 4 pages in a frame)

FIFO Advantages : simple & easy to implement
Dis : Poor Performance

Doesn't consider the frequency of use or last used time, simply replaces the oldest page.

Suffers from Belady's Anomaly (i.e. more page faults when we increase the no. of page frames.

1	1	1	1	5	5	5	5
	2	2	2	2	1	1	1
		3	3	3	3	3	3
			4	4	4	4	4
M	M	M	M	M	M	H	H

5	5	2	2
1	1	1	1
3	3	3	3
6	6	6	6
M	H	M	H

5	5	2	2
1	1	1	1
6	6	6	6
4	3	3	3
M	M	M	H

Total pg. fault = 09

LRU 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	6	6	6	6
M	M	M	M	M	M	H	H	M	H	M	H

Total page fault = 8

In LRU whenever page Replacement happens, the page which has not been used for the longest amount of time is replaced.

Advantages: Efficient
Doesn't suffer from Belady's Anomaly.

DisAdvantages: Complex Implementation.
Expensive
Requires H/W support

Optimal : In this algorithm, pages are Replaced which would not be used for the longest duration of time in the future. i.e. the pages in the memory which are going to be refered farthest in the future are replaced.

T. miss 6

Advantages : Easy to implement.
Simple Data structures are used
Highly efficient.

Disadvantages : Requires future knowledge of the program.
→ Time consuming.

1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3

1	1	1	1	1	1	1	1	6	6	6	6
2	2	2	2	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3	3	3	3
		4	5	5	5	5	5	5	5	5	5
m	m	m	m	m	H	H	H	m	H	H	H

Total page fault = 06

Optimal 4 5 7 12 4 5 13 4 5 7

4	4	4	4	4	4	4	4	4	4
	5	5	5	5	5	5	5	5	5
		7	7	7	7	7	7	7	7
			12	12	12	13	13	13	13
m	m	m	m	H	H	m	H	H	H

Total page faults = 05

Ex FIFO Consider page Reference string 1,3,0,3,5,6,3 with 3 page frames. Find number of page faults for FIFO replacement policies.

1	1	1	1	5	5	5
	3	3	3	3	6	6
		0	0	0	0	3
m	m	m	H	M	m	M

Total page fault = 6

LRV 1, 3, 0, 3, 5, 6, 3

1	1	1	1	5	5	5
0	3	3	3	3	3	3
		0	0	0	6	6
m	m	m	H	M	M	H

Total page fault = 5

optimal 1, 3, 0, 3, 5, 6, 3

1	1	1	1	5	5	5
	3	3	3	3	3	3
		0	0	0	6	6
M	M	M	H	M	M	H

FIFO 4 frame X

1	1	1	1	1	1	1	1	6	6	6
	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3
			4	4	4	4	4	4	4	4
				5	5	5	5	5	5	5
M	M	M	M	M	H	H	H	m	H	

~~6, 0, 5, 2, 0~~

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

3 frames

1	1	1	4	4	4	5	5	5	3	3	3	
	2	2	2	1	1	1	1	1	1	4	4	
		3	3	3	2	2	2	2	2	2	5	
m	m	m	m	m	m	m	m	H	H	m	m	m

---

3 2 1 0 3 2 4 3 2 1 0 4

FIFO

3	3	3	0	0	0	4	4	4	4	4	4	
	2	2	2	3	3	3	3	3	1	1	1	
		1	1	1	2	2	2	2	2	0	0	
M	M	M	M	M	M	M	H	H	M	M	M	
									4	4	4	
									1	1	1	
									2	0	0	
									M	M	H	

(9) pg fault

3	3	3	3	3	3	4	4	4	4	0	0	
	2	2	2	2	2	2	3	3	3	3	4	
		1	1	1	1	1	1	2	2	2	2	
			0	0	0	0	0	0	1	1	1	
m	m	m	m	H	H	m	m	m	m	m	m	

(10) pg faults

Belady's Anomaly

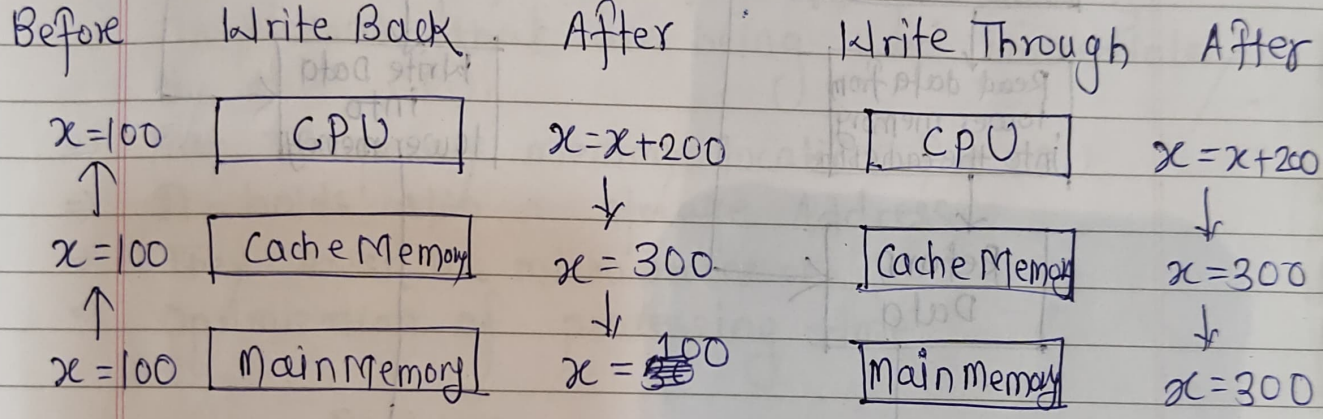
A cache's write policy is the behavior of a cache while performing a write operation.

Hit : write thr', back

Miss : write-allocate, no write allocate.

A device for storing digital information that is fabricated by using integrated circuit technology is known as semiconductor memory.
Thus semiconductor devices are preferred as primary memory.
These are 2 main types of SM technology
non-volatite (ROM)
volatile (RAM) →

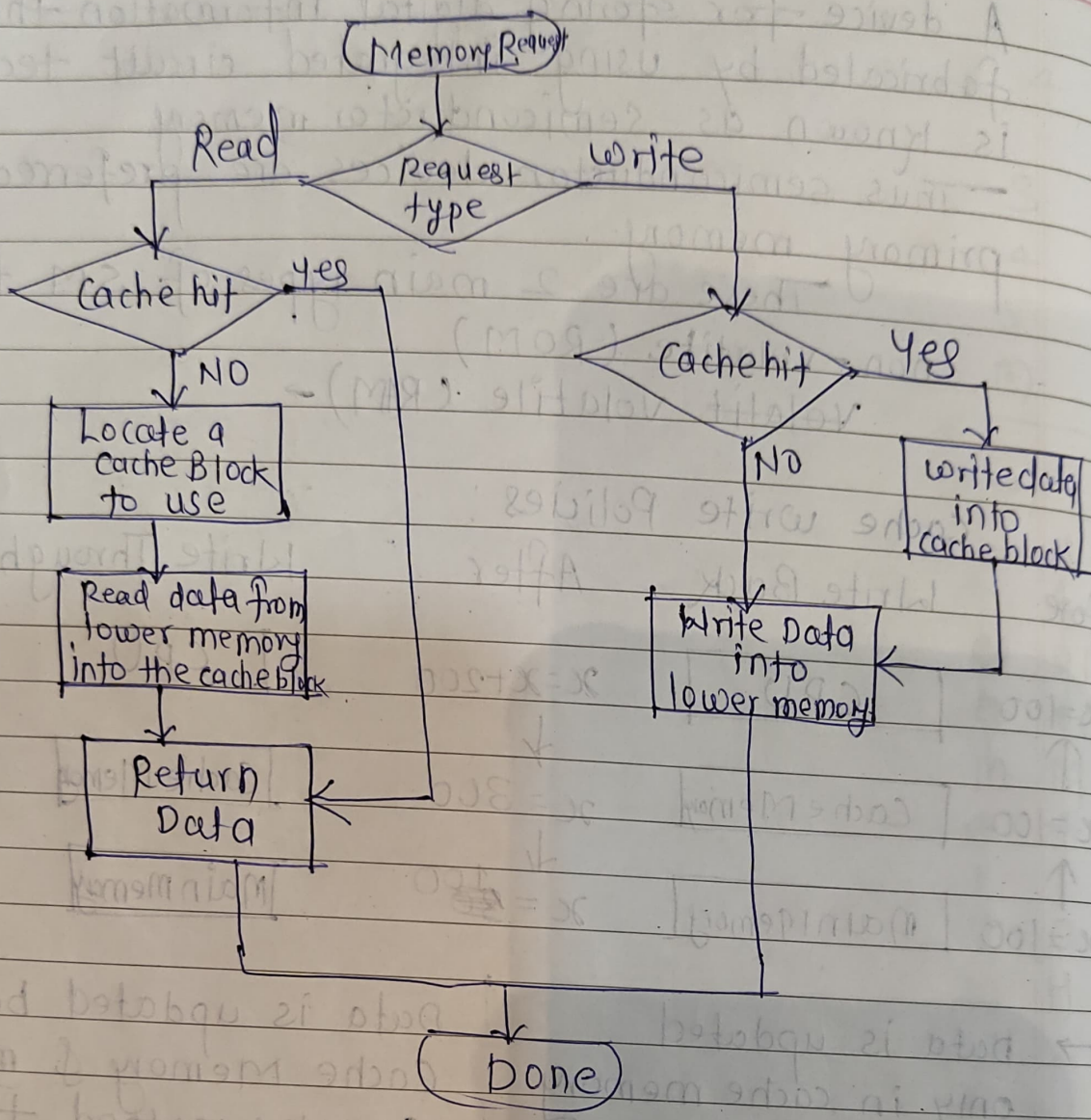
Cache write Policies:



→ Data is updated only in cache memory
→ simple method to implement.
→ When a cache block is not required then selected cache block is written back to main memory to UPDATE the main memory.
→ commonly used method
→ UnReliable Method
→ No Data Redundancy
→ No wastage of time as data occurs only in cache memory.

Data is updated both in Cache Memory & main Memory
complex method to implement.
When cache block is not required there is no need to written back to main memory.

Unpopular Method
Reliable Method
Data Redundancy
Wastage of time as each data updates both in cache & main memory


Memory Request
Request type
Read
Write
Cache hit
yes
NO
Locate a cache Block to use
Read data from lower memory into the cache block
Return Data
Cache hit
yes
NO
write data into cache block
Write Data into lower memory
Done