Module- 4

Characteristics of Memory :

- Location: It deals with the location of the memory device in the computer system. There are 3 possible locations:
  a) CPU: It is often in the form of CPU registers & small amount of cache.
  b) Internal / main: This is the main memory like RAM or ROM. The CPU can directly access the main memory.
  c) External / Secondary: It comprises of secondary storage devices like hard disks, magnetic tapes. The CPU doesn't access these devices directly. It uses device controllers to access secondary storage devices.

- Capacity: The capacity of any memory device is expressed in terms of:
  a) Word Size: Words are expressed in bytes (8 bits). A word can however mean any number of bytes. Commonly used word

sizes are 1 byte (8 bits), 2 bytes (16 bits) and 4 bytes (32 bits).

b) Number of words: This specifies the number of words available in the particular memory device. For example, if a memory device is given as 4k × 16. This means the device has a word size of 16 bits & a total of 4096 (4k) words in memory.

- Unit of transfer: It is the maximum number of bits that can be read or written into the memory at a time. In case of main memory, it is mostly equal to word size. In case of external memory, unit of transfer is not limited to the word size; it is often larger and is referred to as blocks. In other words, it is the number of data lines into or out of the memory module.

Addressable unit: Smallest location which can be uniquely addressed.

| ✱ Word: The natural unit of organisation of memory. |

- Access methods: It is a fundamental characteristic of memory devices. It is the sequential sequence or order in which memory can be accessed. There are 4 types of access

methods:

a) Sequential: Simplest access method. Information in the file is processed in order, one record after the other. The most common mode of access. Access time depends on location of data and previous location.
eg: tape.

b) Direct: (or relative access method), A fixed length logical record that allows the program to read & write record rapidly, with no particular order. There is no restriction on the order of reading & writing.
Access time depends on location & previous location. eg: Disk.
files are viewed as a numbered sequence of block or record.

* file record: A collection of related fields of information.
* file block: fixed length chunks of data that are read into memory when requested by an application.

c) Random: If storage locations of a particular memory device can be accessed in any order and access time is independant of the memory location being accessed, such memory devices are said to have a random access mechanism.

eg: RAM.

d) Associative access: In this memory, a word is accessed rather ~~that~~ than its address. This access method is a special type of ~~rara~~ random access method. Also independent of ~~loca~~ location.
eg: Cache.

- Performance: The performance of the memory system is determined using three parameters
  a) Access time: In random access memories, it is the time taken by memory to complete the read / write operation from the instant that an address is sent to the memory. For non-random access memories, it is the time taken to position the read write head at the desired location. Access time is widely used to measure performance of memory devices.

  b) Memory cycle time: It is defined only for RAMs and is the sum of the access times & the additional time required before the second access can commence. (time blw two consecutive read requests).

  c) Transfer rate: It is defined as the rate at which data can be transferred into or out of a memory device.

- Physical type: Memory devices can be either semiconductor memory, magnetic memory or optical memory
(Primary)
a) Semiconductor memory: have faster access time, smaller storage capacity, & higher cost per bit of storage, as compared to secondary memory. (RAM, ROM, etc.)
b) Magnetic memory (Secondary) & optical memory (Secondary): Slow compared to semiconductor memory. But they are cheaper than semiconductor memory. Not static devices.
eg of magnetic: hard disk; magnetic disk & tape.
eg of optical: CD, DVD.

- Physical Characteristics:
a) Volatile / Non-Volatile: If a memory device continues to hold data even if power is turned off, the memory device is non-volatile, else it is volatile.
b) Erasable / Non-Erasable: The memories in which data once programmed cannot be erased are called non-erasable memories. Memory devices in which data in the memory can be erased is called erasable memory.
erasable → RAM
non-erasable → ROM

c) Decay: Memory fade that happens due to time (?)

d) Power consumption: Some types of memories take more power, eg: DRAM > SRAM

- Organisation: The physical arrangement of bits to form words. The obvious arrangement is not always used.

Memory heirarchy:

- Registers: The register is usually a static RAM in the computer processor that is used to hold the data word that is typically 64 bits or 128 bits. A majority of the processors make use of a status word register and an accumulator. The accumulator is primarily used to store the data in the form of mathematical operations & the status word register is primarily used for decision making.

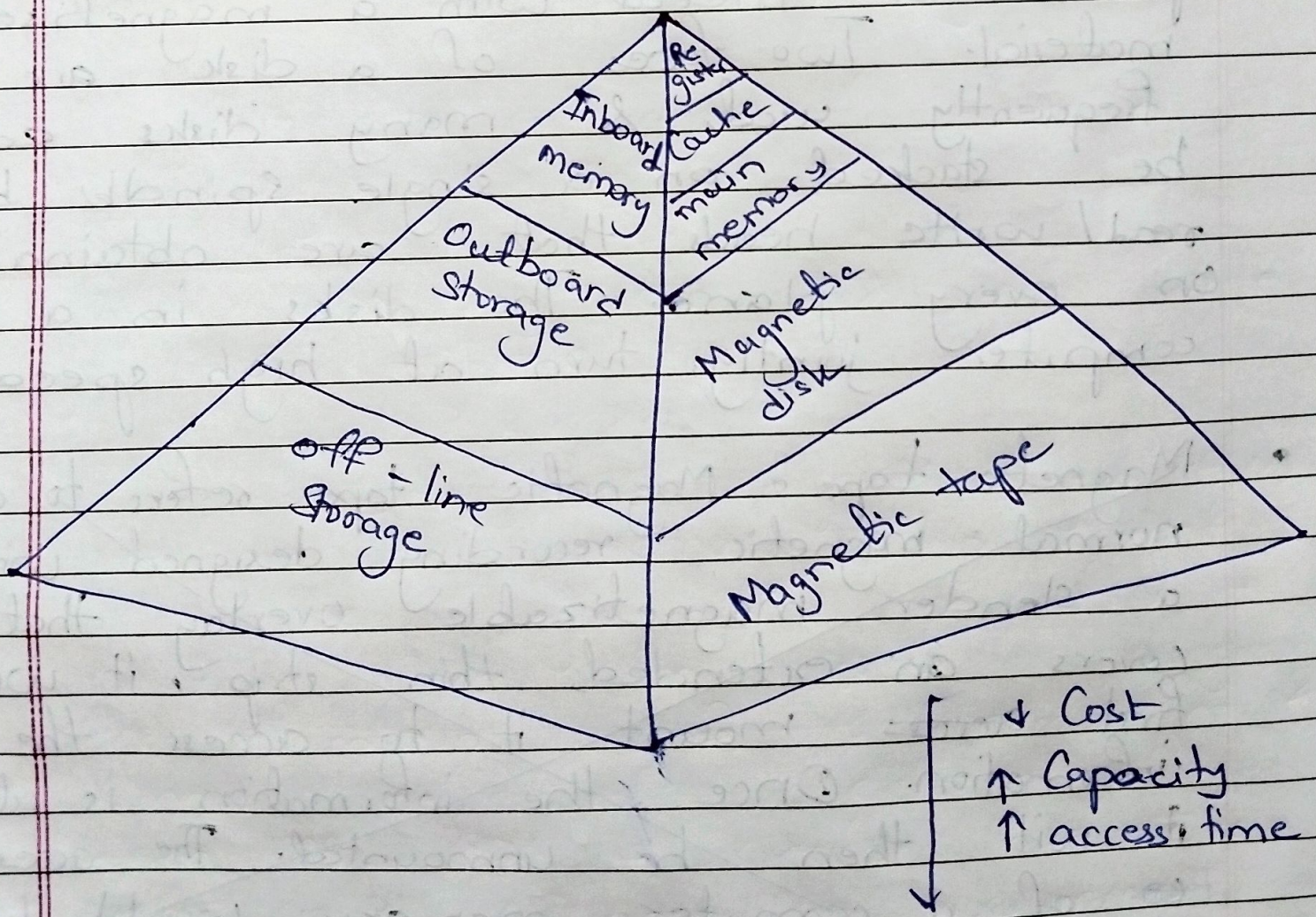
- Cache memory: The cache basically holds a chunk of information that is used frequently from the main memory. We can also find cache memory in the processor. In case the processor has a single-core, it will rarely have multiple cache levels. The present multi-core processors would have 2 levels for every individual core, & one of the levels is

shared.

- Main memory: The CPU's memory unit that communicates directly. It's the primary storage unit of a computer system. The main memory is very fast & a very large memory that is used for storing the information throughout the computer's operations. This type of memory is made up of RAM as well as ROM.

- Magnetic Disks: The magnetic disks are circular plates that's fabricated with plastic or metal with a magnetised material. Two faces of a disk are frequently used, & many disks can be stacked on a single spindle by read/write heads that are obtainable on every plane. The disks in a computer jointly turn at high speed.

- Magnetic tape: A normal magnetic recording designed with a slender magnetizable overlay that covers an extended thin strip of plastic film. It is used mainly to back up huge chunks of data. When a computer needs to access a strip, it will first mount it to access the information. Once the information is allowed, it will then be unmounted. The actual access time of a computer memory would be slower within a magnetic strip, & it will take a few minutes for us to access a strip.



PAGE NO.
DATE / /

RAM

- Dynamic RAM ~~(DRAM)~~ (DRAM): Memory is made up of bits of data or program code that are arranged in a two dimensional grid. DRAM will store bits of data in what's called a storage, or memory cell, consisting of a capacitor and a transistor. A DRAM storage cell is dynamic; meaning that it needs to be refreshed or given a new electronic charge every few milliseconds to compensate for charge leaks from the capacitor.
DRAM is a type of semiconductor memory (Primary memory).

- Static RAM (SRAM): holds data as long as the memory has power. Does not need to be refreshed. SRAM stores a bit of data on four transiters using 2 cross coupled inverters.
Uses latching circuitry (flip-flop) to store each bit. SRAM is volatile memory, data is lost when power is removed.

-

SRAM	DRAM
a) ~~Both~~ volatile, as data is lost when power is removed.	a) Volatile, as data is lost when power is removed.

SRAM	DRAM
b) More difficult to build	b) Simpler to build
c) Bigger	c) Smaller
d) less dense	d) More dense
e) More expensive	e) less expensive
f) Needs no refresh	f) Needs refresh
g) Smaller memory units	g) larger memory units.
h) faster	h) Slower
i) Cache	i) Main Memory
j) Digital	k) ~~Ato~~ Analog

(Refer to diagrams of both in ppt pages 18 & 20)

ROM : Can only read ROM, not write.
Non- volatile

- Programmable Read only Memory (PROM): A computer memory chip that can be programmed once after it is created. Once the PROM is programmed, the information written is permanent & cannot be erased or deleted.
When the PROM is created, all bits read as "1". During the programming, any bit needing to be changed to a '0' is etched or burned into the chip. using

Non - volatile, writing performed electrically at the time of chip fabrication.

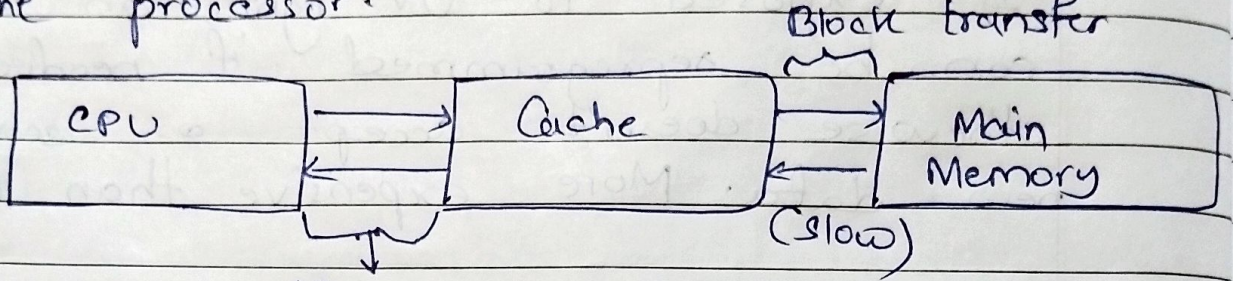
- EPROM: Erasable PROM, non-volatile memory chip that can only be read. If exposed to UV light, an EPROM can be reprogrammed if needed, but otherwise doesn't accept or save any new data. More expensive than PROM.

- EEPROM: Electrically EPROM, is a prom that can be erased and reprogrammed using an electric charge. More expensive than EPROM.

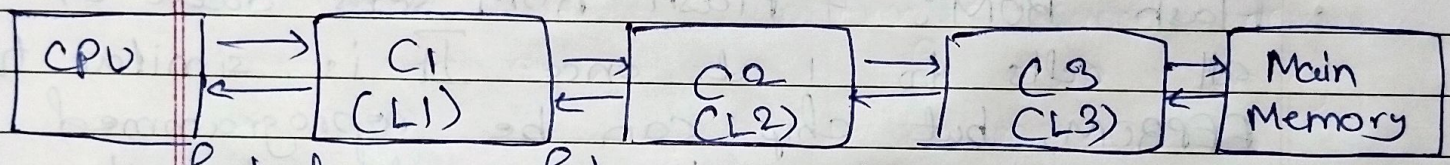
- Flash ROM: A Flash ROM sets data of all cells to 1 at once. It is similar to EEPROM, but chip can be reprogrammed while still inside the computer; thus its easier to upgrade programs stored in Flash ROM. Access time is 40 - 90 ns. used in modems.

Cache: Cache memory is intended to give memory speed approaching that of the fastest memory available, & at the same time provide a large memory size at the price of less expensive types of semiconductor memories.
Cache delivers the word that CPU wants to read from memory. It is located on CPU chip or module.

If the coord that CPU wants is not in caches a block of main memory, consisting of some fixed ~~coords~~ number of words is read into the cache & then the word is delivered to the processor.



(Single Cache)



Size : L3 > L2 > L1

Cache memory is a high speed memory kept in between processor & RAM to increase the data execution speed. It is kept near the processor.

→ Main memory consists of up to 2^n addressable ~~use~~ words.

→ Each word has a unique n - bit address.

→ Memory consists of a number of fixed length blocks of k words each.

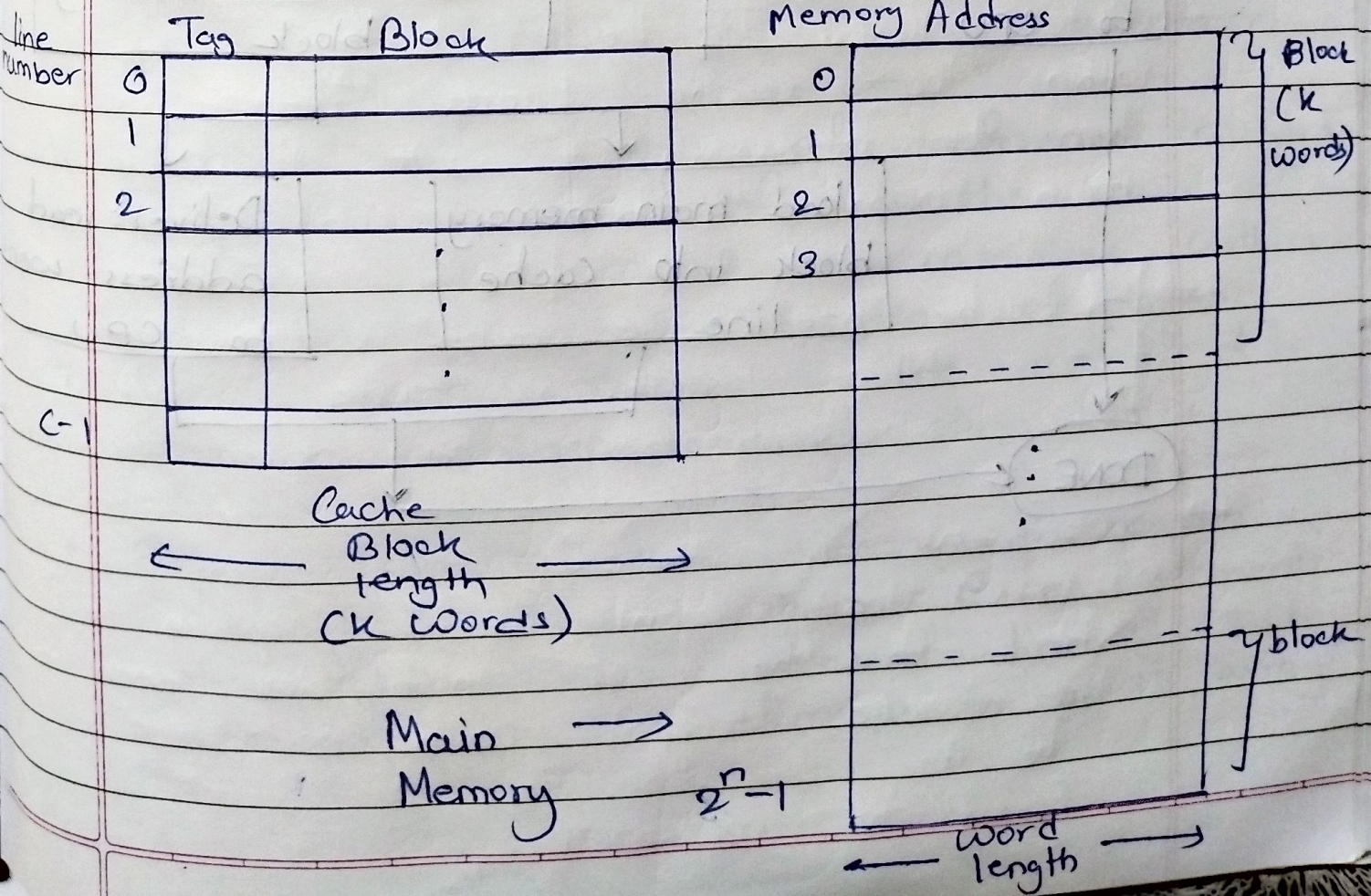
→ Main memory blocks (M) = $\frac{2^n}{k}$

→ Cache blocks = m, also called lines.

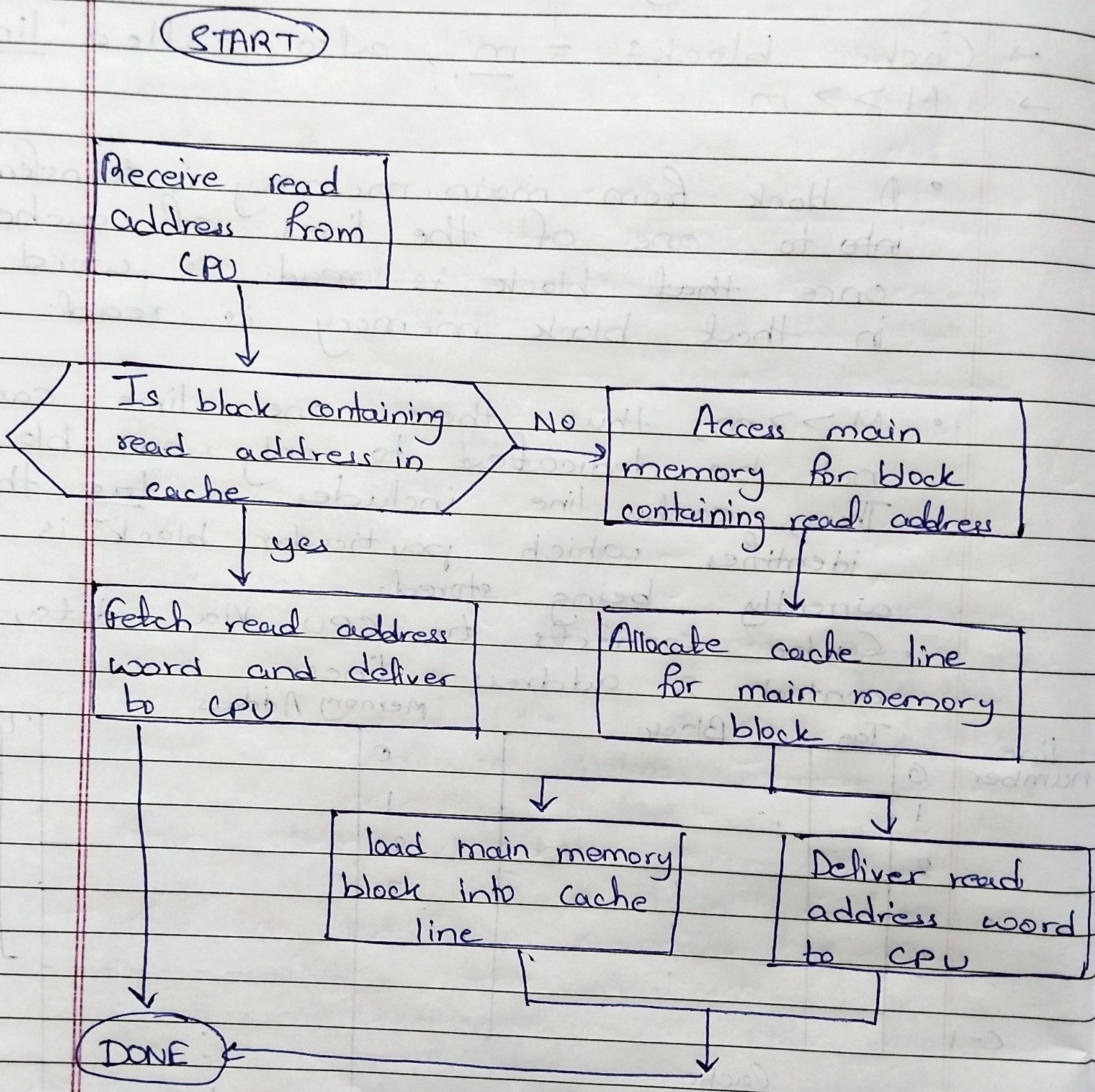
→ M >> m

- A block from main memory is transferred into to one of the lines of cache once ~~that block is read~~. a word in that block memory is read.

- M >> m, thus there one line can not be dedicated to only one block. Thus, each line includes a tag that identifies which particular block is currently being stored.
- Cache connects to CPU via data, control & address lines.

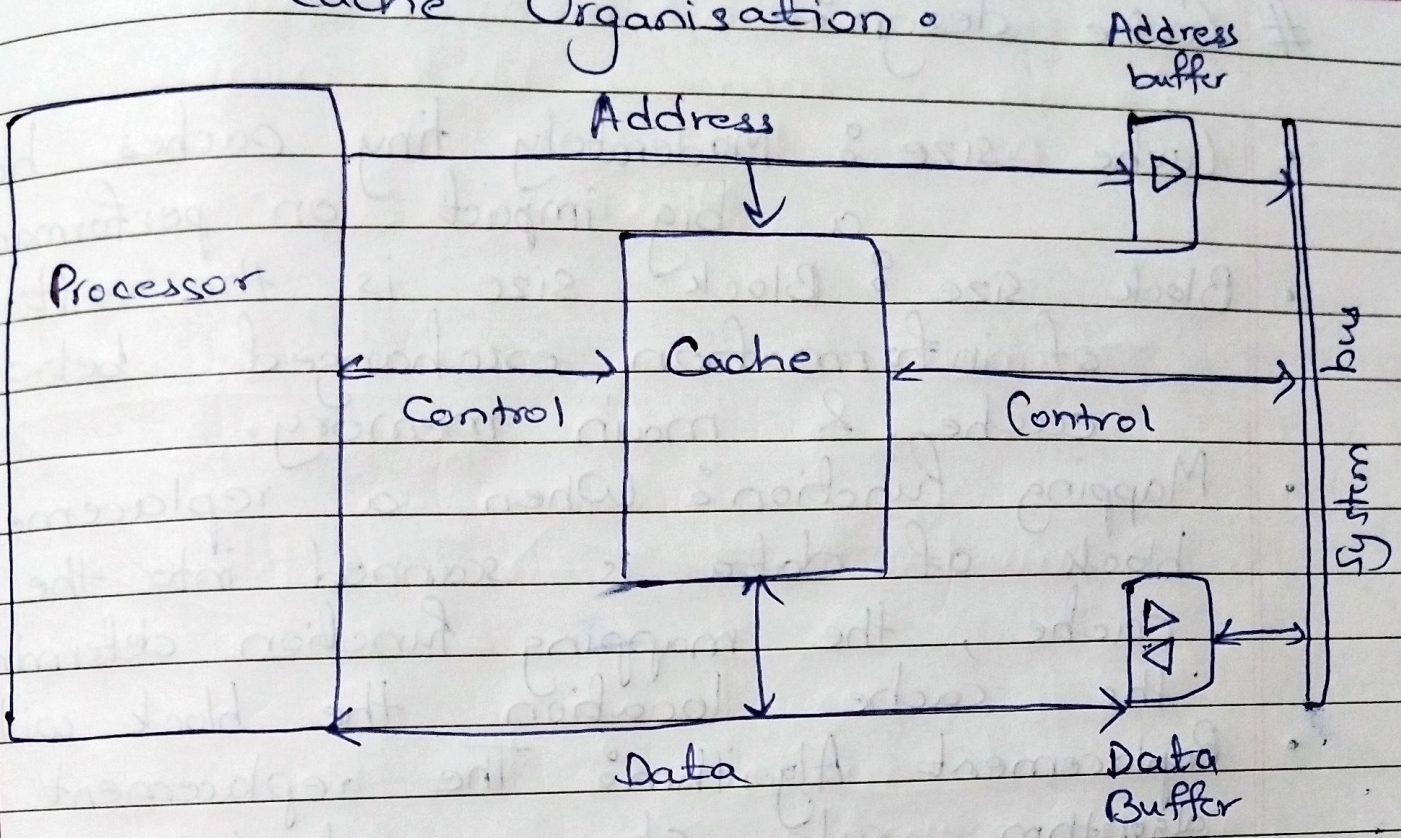


Cache Read Operations



DATE / /

Cache Organisation:



locality of reference: The tendancy of a processor to access the same set of memory locations repititively over a short period of time, thus forming a cluster of memory references.

There are 3 types of reference locality:

- Temporal locality: refers to the reuse of specific data and/or resources within a relatively small duration of time.
- Spacial locality: refers to the use of data elements within relatively close storage locations.
- Sequential locality: special case of spacial locality, occurs when data elements are arranged & accessed linearly, such as traversing the elements in a 1D array.

Cache design:

- Cache size: Moderately tiny caches have a big impact on performance.
- Block size: Block size is the unit of information exchanged between cache & main memory.
- Mapping function: When a replacement block of data is scanned into the cache, the mapping function determines the cache location the block will occupy.
- Replacement Algorithm: The replacement algorithm rule chooses, at intervals, the constraints of the mapping function and which block to interchange once a replacement block is to be loaded into the cache.
- Write policy: If the contents of a block within the cache is altered, then its necessary to write it back to main memory before exchanging it.

→ ✱ Cache size: More cache is expensive
More cache is faster
Checking cache for data takes time

→ Replacement Algorithms:

a) (LRU) least recently used: Replace that block in the set that has been in the cache longest with no reference to it.
Easy to implement.

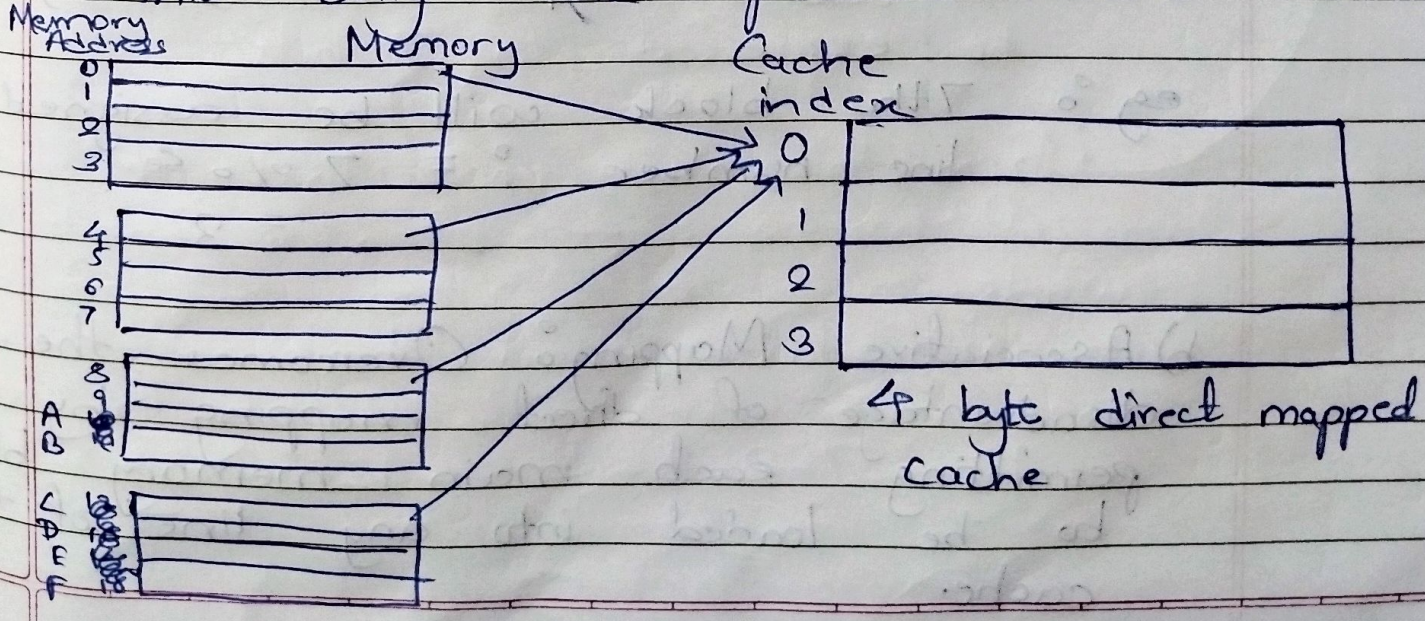
b) First in first out (FIFO): Replace that block in the set that has been in the cache the longest.

c) Least frequently used (LFU): Replace that block in the set that has experienced fewest references.

d) Random: pick a line at random from among the candidate lines.

→ Mapping Techniques: Since there are fewer lines than blocks (M >> m), an algorithm is needed for mapping blocks into cache lines. First, we need a way to determine which main memory block currently occupies a cache line.

a) Direct Mapping: The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line.



i = cache line member number
j = main memory block number
m = no. of lines in the cache

$$i = j \bmod m \quad (j \,\%\, m)$$

→ Formula for tag = $\frac{\text{no. of blocks in main.m}}{\text{no. of lines in cache}}$

→ Formula for SET ⇒ If block set has of lines, then

In direct mapping, the blocks of the main memory are assigned to lines of the cache as follows:

Cache line	Main memory blocks assigned
0	0, m, 2m, ...
1	1, m+1, 2m+1, ...
:	:
m-1	m-1, 2m-1, 3m-1, ...

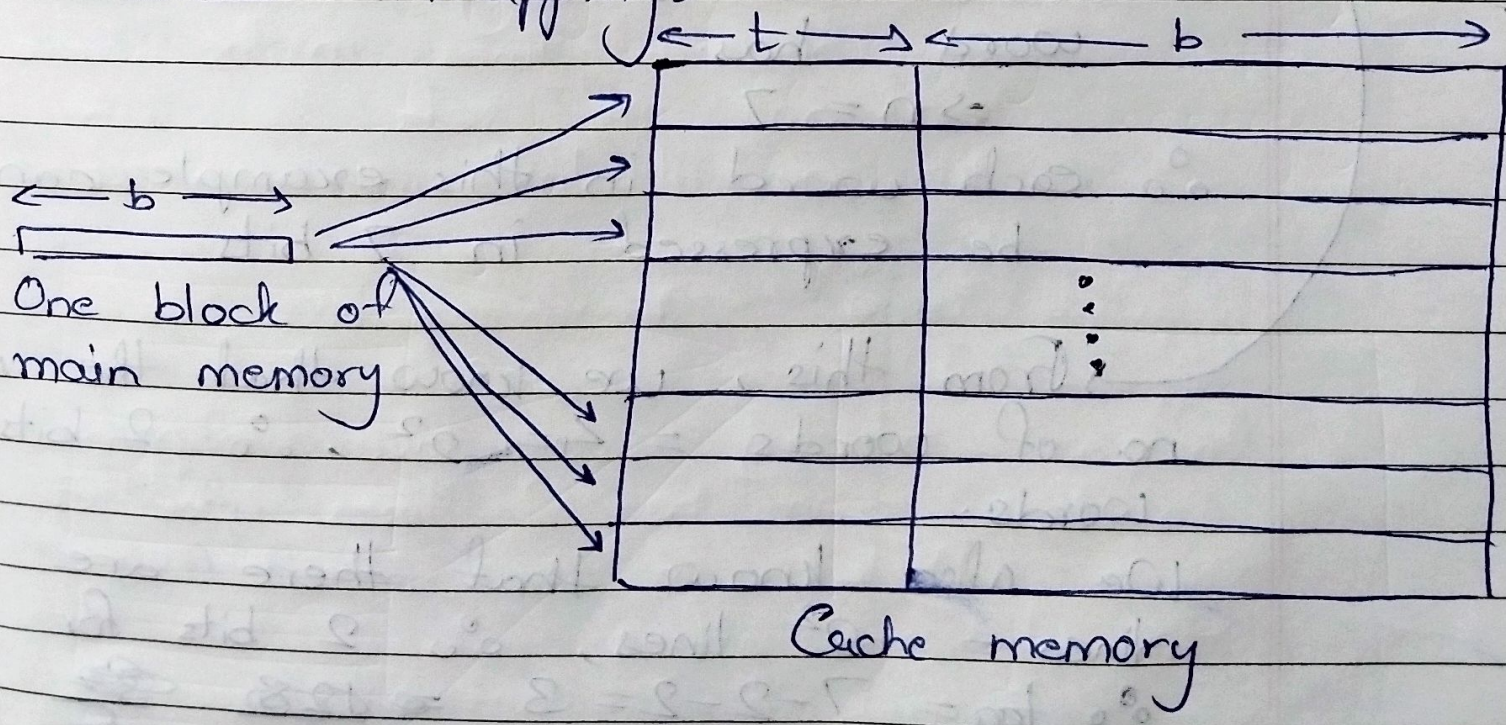
eg: 7th block will be assigned to line number $i = 7 \,\%\, 4$
$= 3$

b) Associative Mapping: Overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache.

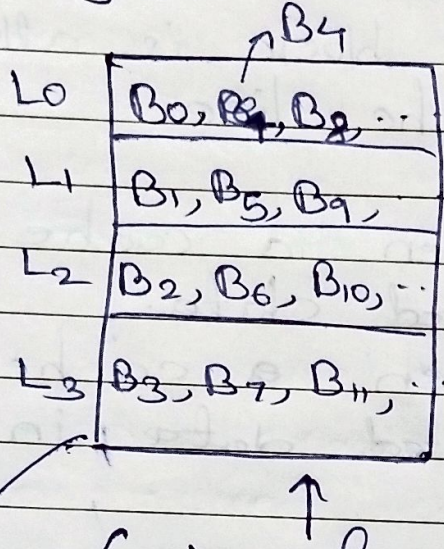
* The disadvantage of direct mapping was that there were more chances of some cache lines being empty, hence more chances of 'miss' than 'hit'. This is overcome in associative mapping as any block is allowed in an empty cache line.

* 'Hit' means when a cache successfully finds the requested data.
'Miss' means when a cache doesn't have the requested data in the memory.

* Note that a cache line can only contain one block, but there are 4 possible block numbers that can be assigned to that cache line (for direct mapping).



- lets, for example, say that there are 4 cache lines and 32 main memory blocks, with each block containing 4 words. (1 word can be any number of bytes) ∴



W_0	W_1	W_2	W_3	B_0
W_4	W_5	W_6	W_7	B_1
8	9	10	11	B_2
12	13	14	15	B_3
120	121	122	123	B_{30}
124	125	126	127	B_{31}

128 words (4 × 32)

Using formula ($i = j \% m$) for direct mapping

no. of words = $128 = 2^n$

n = no. of bits that each word has.

$\Rightarrow n = 7$

∴ each word in this example can be expressed in 7 bits

From this, we know that the max no of words = $4 = 2^2$, ∴ 2 bits for words.

We also know that there are 4 lines = 2^2 lines, ∴ 2 bits for line no.

∴ tag = 7 − 2 − 2 = 3

DATE / /

∴ for above example, an address bits represent ∴

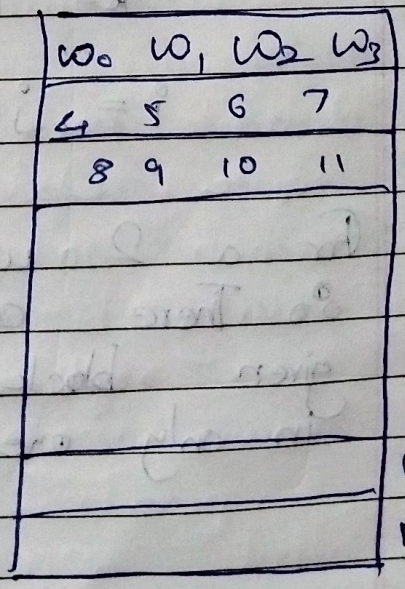
Tag	No. of lines	Words
3	2	2

∴ First 3 bits of an address will represent the tag, the next two will represent the no. of the line, & the last two bits will give the no. (or index) of word.
Thus, with the address of a word, it will be located using above method.
eg∴ lets say word's address = 00001010
(8 bits.)

tag = 000
line no. (index) = 10 = 2
word no. (index) = 10 = 2
this is confirmed by the above diagram, B_{10} is located at line index 2 & word index 2.

For Associative Mapping ∴

L_0	$B_1, B_0, B_2, \ldots$
L_1	$B_0, B_1, B_2, \ldots$
L_2	$B_0, B_1, B_2, \ldots$
L_3	$B_0, B_1, B_2, \ldots$



fixed line no.

Fully Associative has no ~~tag~~ because any block can be assigned to any line. Thus, continuing previous example of 7 bit address, words will take 2 bits still, and tag will be the remaining 5 bits

Tag	Word
5	2

c) Set Associative mapping : Set Associative mapping exhibits the strengths of both the direct and associative approaches while reducing their disadvantages. The cache consists of a number of sets, each of which consits of a number of lines.

i = cache set number
j = main memory block number
m = no. of lines in the cache
v = no. of sets
k = no. of lines in each set.

$i = j \% v$

For a 2 way associative mapping, k = 2. So There can be 2 lines per set. A given block can be in one of 2 lines in only one set.

Formulas :

① Tag size = $\log_2 \frac{\text{No. of blocks in M.M}}{\text{No. of sets in cache}}$

② No. of lines in each cache set = $\frac{\text{Cache Size}}{\text{no. of ways}}$

③ Set Size = $\log_2$ (no. of lines in Cache)

④ Word length = $\log_2$ (No. of words in a block)

⑤ Size of the main memory
= No. of blocks × Size of each block

(Solve all the numericals in ppt page 58)

Comparison :

Direct Mapping	Associative Mapping
a) Simple	a) Not so simple IG
b) Inexpensive	b) Cache searching gets expensiv
c) Fixed location of a given block	c) location not fixed for a block.
d) High chances of misses	d) lower chances of misses.
e) no. of line required for pinpoint block	e) Tag & word index are enough.

→ Write policy: When a block that is resident in the cache is to be replaced, there are two cases to consider. If the old block in the cache has not been altered, then it may be overwritten with a new block without first writing out the old block. If at least one write operation has been performed on a word in that line of the cache, then main memory must be updated.
There are 3 ways to do so:

a) ~~Writh~~ + Write through: When a block is updated, all write is sent to cache as well as main memory, ensuring that main memory is always valid. Multiple CPUs or I/O devices can keep their local cache up to date using main memory. Disadvantage is that a lot of traffic is created, write is also slowed down.

b) Write back: ~~Minimizes~~ Minimizes memory write. Updates are made only in cache. When an update occurs, an update bit/ dirty bit is set. Then, when a block is replaced, it is written back to the main memory if & only if the update bit is set. Disadvantage is that main memory is invalid. Hence, accesses by I/O modules can only be allowed

through the cache.

c) Buffered write through: A variation of write through where the cache uses a "write buffer" to hold data being written back to main memory. This frees the cache to service read requests while the write is taking place. There is usually only one stage of buffering so subsequent writes must wait until the first is complete.
Basically, read access can be performed simultaneously, write operations can't.

- (I forgot a point under Cache design)
No. of Caches: When caches were originally introduced, the typical system had a single cache. More recently, the use of multiple caches has become the norm. ~~Two aspects of this design issue concern the number of levels of caches.~~

Multilevel caches: It has become possible to have a cache on the same chip as the processor: on-chip cache. Compared with a cache reachable via an external bus, the on chip cache reduces the processor's external bus activity & therefore speeds up execution times & increases overall system performance. This internal cache is Level 1 (L1), the typical range for size is 8 kB to 64 kB & it uses high speed SRAM instead of slower DRAM.

An L2 cache is also included because if there is no L2 cache & the processor makes an access request for a memory location not in L1 cache, then the processor must access DRAM or ROM across the bus, which is slow. If an L2 SRAM is used, then information can be frequently received. Typical range for size is 64KB to 4MB.

L3 caches are found on the motherboard rather than the processor. It is kept b/w ram and L2 cache.

Cache coherence: Because each local cache contains an image of a portion of memory, if a word is altered in one cache, it could conceivably & invalidate a word in another cache. This problem is known as cache coherence. We have following solutions to deal with this problem:

a) Software solution: Software cache coherence schemes attempt to avoid the need for additional hardware circuitry & logic by relying on the compiler & operating system to deal with the problem. Software approaches are attractive because the overhead of detecting potential problems is transferred from run time to compile time, & the design complexity is transferred from hardware to software.

The simplest approach is to prevent any shared data variables from being cached.

b) Hardware Solutions: Generally referred to as cache coherence protocol. These solutions provide dynamic recognition at run time of potential inconsistency conditions. Because the problem is only dealt with when it actually arises, there is more effective use of caches, leading to improved performance over a software approach. These approaches are transparent to the programmer & the compiler.
Snoopy protocols are also hardware schemes.

c) Snoopy protocol: Distribute the responsibility for maintaining cache coherence among all of the cache controllers in a multiprocessor. A cache must recognise when a line that it holds is shared with other caches. When an update action is performed on a shared cache line, it must be announced to all other caches by a broadcast mechanism. Each cache controller is able to "Snoop" in on the network to observe these broadcasted notifications. Suited to bus based multiprocessor because the shared bus provides a simple means for broadcasting & snooping.

Snoop → investigate

Disadvantge is that the bus traffic is increased.

There are 2 types of basic approaches to snoopy protocol: Write update & Write invalidate.

d) Write update: There can be multiple writers as well as multiple readers. When a processor wishes to update a shared line, the word to be updated is distributed to all others, & caches containing that line can update it.

e) Write invalidate: There can be multiple readers but only one writer at a time. Initially, a line may be shared among several caches for reading purposes. When one cache wants to perform a write to the line, it first issues a notice that invalidates that line in the other caches, making the line exclusive to the writing cache. Once the line is exclusive, the owning processor can make cheap local writes until some other processors requires the same line.

f) The MESI Protocol:

M, E, S and I are states

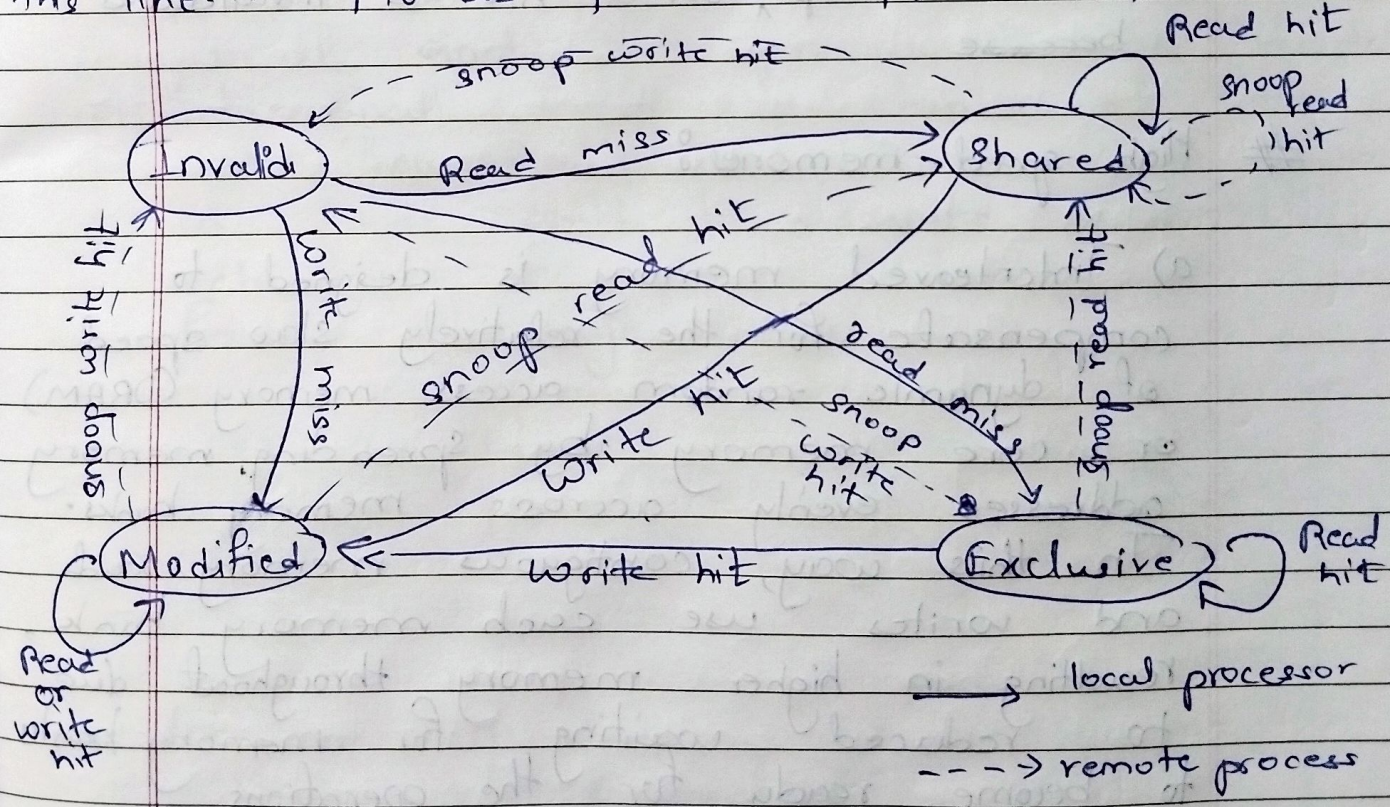
M: Modified, The line in the cache has been modified (different from main memory) and is available in only this cache

E: Exclusive, The line in the cache is the same as that in main memory and is not present in any other cache.

S: Shared The line in the cache is the same as that in main memory & may be present in any other cache.

I: Invalid, The line in the cache does not contain valid data.

Is	M	E	S	I
This cache line valid?	Yes	Yes	Yes	No
The memory copy is...	Out of date	Valid	Valid	—
Copies exist in other caches?	No	No	maybe	maybe
A write to this line...	doesn't go to bus	doesn't go to bus	goes to bus & updates cache	goes directly to bus.



i) Modified snoop write hit is invalid because main memory needs to be updated before it can be written.

ii) Exclusive snoop read hit becomes shared because that remote process has now accessed the memory.
local

iii) Shared write hit becomes modified because once you write in a shared data, you'll have to invalidate all other caches with which data was shared.

iv) Modified snoop read hit becomes shared because external caches have now read the data.
(or remote)

v) Shared snoop write hit is invalidated ~~because~~

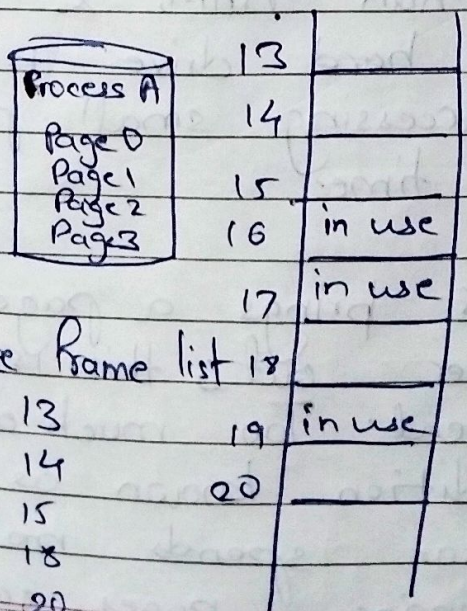
High speed memories:

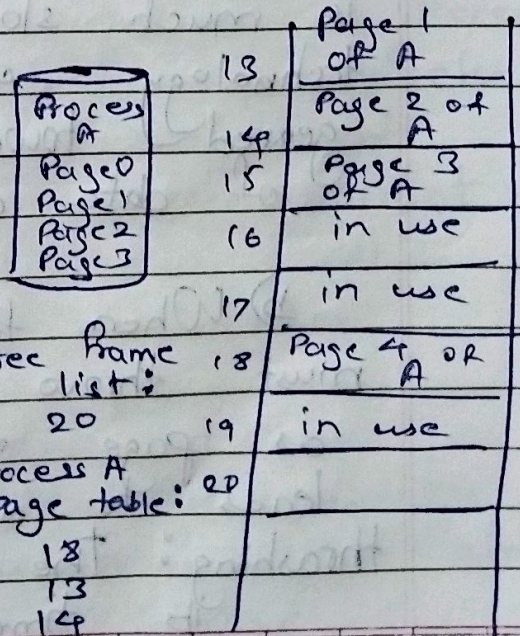
a) Interleaved memory is designed to compensate for the relatively slow speed of dynamic random access memory (DRAM) or core memory by spreading memory addresses evenly across memory banks. In this way, contiguous memory reads and writes use each memory bank, resulting in higher memory throughput due to reduced waiting for memory banks to become ready for the operations.

b) Associative memory: Also known as Content Addressable Memory (CAM), can be considered as a memory unit whose stored data can ~~by~~ be identified for access by the content of the data itself rather ~~that~~ than by an address or memory location.
Reference Clues are associated with actual memory contents until a desirable match (or set of matches) is found.

Memory Management:

- Paging: Memory is partitioned into equal fixed size chunks that are relatively small, and each process is also divided into small fixed size chunks of some size. Then the chunks of a program, known as pages, could be assigned to available chunks of memory, known as frames or page frames.





The page table shows the frame location for each page of the process.

- Virtual memory: works on the concept of demand paging, which simply means that each page of a process is brought in only when it is needed, that is, on demand. At one time, only a few pages of any given memory process are in memory, & therefore more processes can be maintained in memory.
Areas in the RAM that have not been recently used are copied on to the hard disk. This frees up space in RAM to load new application. The copying to hard disk happens automatically. Also, hard disks are much cheaper than RAM chips.
Disadvantages are:
a) Read/Write speed of hard drive is much slower than RAM, & the technology of a hard drive is not geared toward accessing small pieces of data at a time.

b) When the OS brings a page it must throw another out; this is known as page replacement. Too much of this leads to a condition known as thrashing: the processor spends most of its time swapping pages rather

than executing instructions.

- Segmentation: Paging is invisible to the programmer & serves the purpose of providing the programmer with larger address space, segmentation is usually visible to the programmer ~~with a larger address space~~ and is provided as a convenience for organizing programs & data and as a means for associating privilege & protection attributes with instructions & data.
  Segments are of variable (dynamic) size. The program is divided into modules/segments. Not all segments of a process are loaded at a time; nor are they contiguous memory blocks. Segment table is also required.

Main memory Allocation: Memory is divided into set of contiguous locations called regions / segments / pages. It stores blocks of data. Placement of blocks of information in memory is called Memory Allocation. Memory management systems keeps information in a table containing available & free slots.

Secondary storage devices:
a) Floppy disk: used on a computer to store ~~external data~~ data externally.

b) Hard disk: Store & retrieves data using magnetic storage.

c) Magnetic disk: data is stored by modifying or rearranging the magnetism of tiny iron-based magnetic particles present on the band of the disk.

d) Magnetic tape: A medium for magnetic recording, made of a thin magnetisable coating on a long, narrow strip of plastic film.

e) Optical Memory: or secondary memory is non-volatile, cheaper than primary memory and has large capacity.

f) CD-ROM

g) DVD

h) RAID (Redundant Array of Independent disks): A techique which makes use of multiple disks instead of using a single disk for increased performance, data redundancy or both. Data redundancy adds to disk reliability. It acts as backup in case disk failure occurs. If data is spread across multiple disks without RAID, the loss of a single disk can affect the entire data. The raid scheme consists of 6 levels. These levels do not imply a hierarchical relationship but designate different design architectures that share 3 common characteristics:

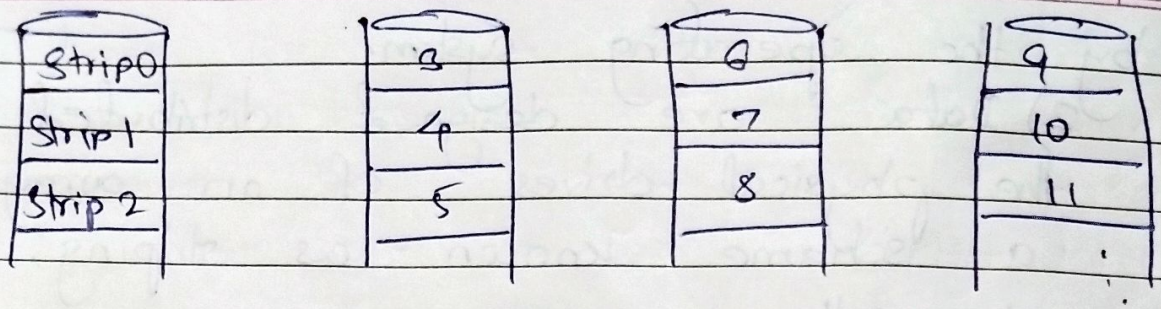
1) RAID is a set of physical disk drives viewed as a single local drive by the

by the operating system.

2) Data are designed distributed across the physical drives of an array in a scheme known as striping, described subsequently.

3) Redundant disk capacity is used to store parity information, which guarantees data recoverability in case of a disk failure.

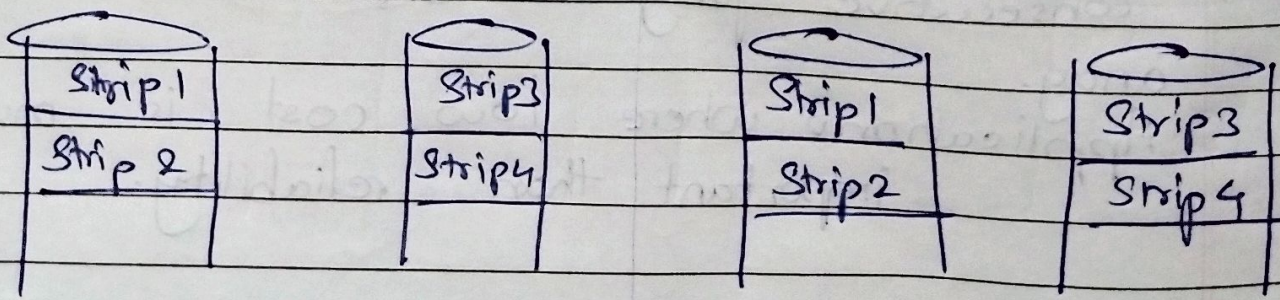
i) Raid level 0: The user and system data are distributed across all of the disks in the array. If two different I/O requests are pending for two different blocks of data, then there is a good chance that the requested blocks are on different disks. Thus, the two requests can be issued in parallel, reducing the I/O queuing time. Does not include redundancy (isn't quite of the RAID family).
User & system data are stored on a logical disk. The logical disk is divided into strips, these strips may be block, sectors or any other unit. The strips are mapped round robin (?) to consecutive physical disks in the raid array.
Application: where low cost is more important than reliability.



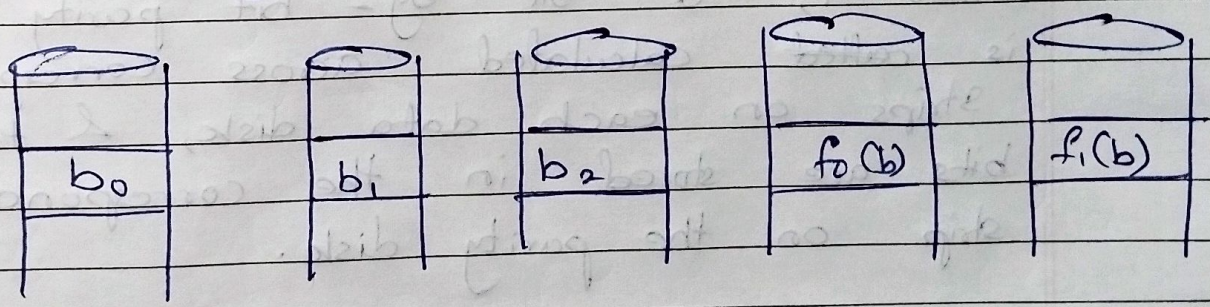
ii) RAID 1: Redundancy is achieved by the ~~simple~~ simply duplicating the data. Data striping is used here too. Each logical ~~stripped~~ strip is mapped (copied) to two separate physical disks so that every disk in the array has a mirror disk that contains the same data. Read can be done from ~~both~~ either one of them but when new data is to be written, both have to be updated. Recovery from a failure is simple. When a drive fails, the data may still be accessed from the second drive.

This RAID is expensive because it requires twice the disk space of the logical disk that it supports.

In transaction-oriented environments RAID 1 is effective if bulk requests are READ.

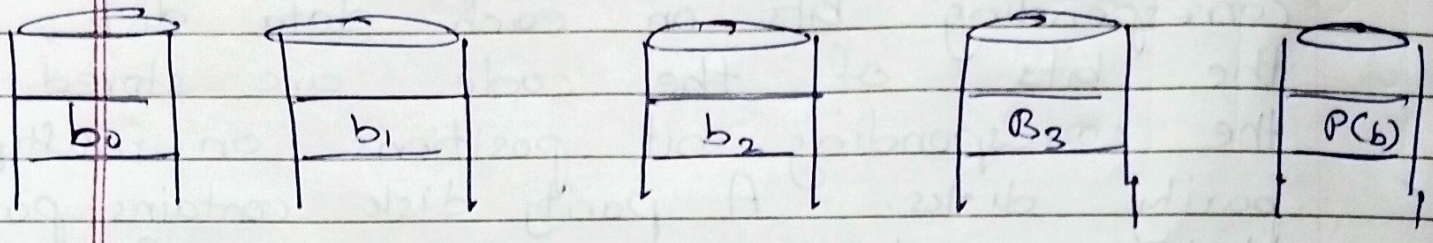


iii) Raid 2: All members of the disks participate in the execution of every I/O request. Thus, the disks are synchronised. The strips are very small, often as small as a single byte or word. A error correcting code is calculated across corresponding bits on each data disk, & the bits of the code are stored in the corresponding bit positions on multiple parity disks. A parity disk contains parity bits; parity bits are a simple form of error detecting code which check for errors, also called er check bit. The above error code is called the hamming code.
RAID 2 is also costly, there are many redundant disks. containing hamming codes. RAID 2 is an effective choice in an environment where many disk errors occur.

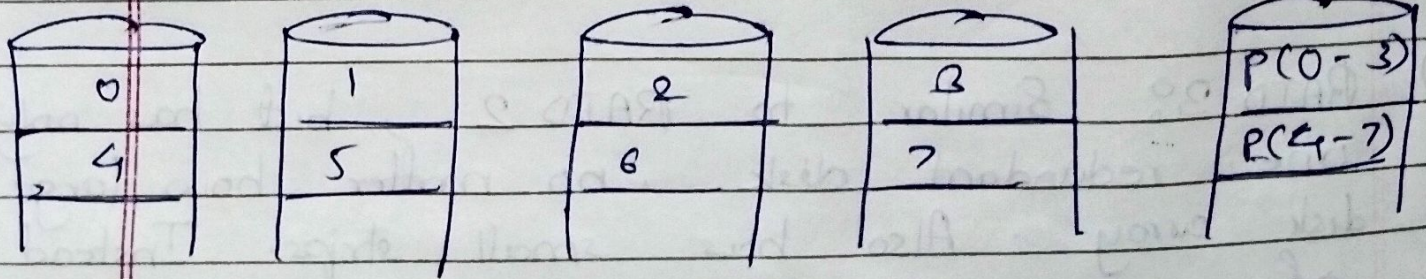


iv) RAID 3: Similar to RAID 2, but has only one redundant disk, no matter how large the disk array. Also has small strips. Instead of error code, a simple parity bit is used to check each set of corresponding bits. In the event of a disk failure, the parity

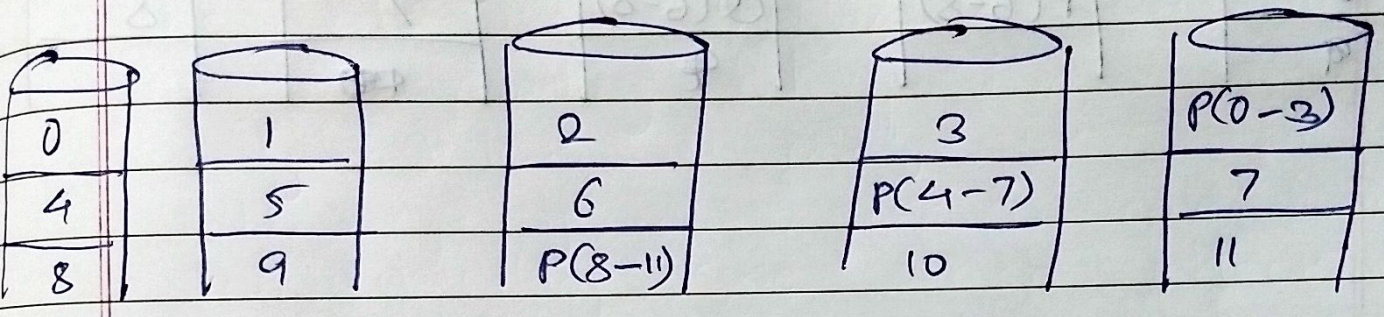
drive is accessed & data is reconstructed from the remaining devices. RAID 3 can achieve very high data transfer rates, thus good to use for ~~high~~ large transfers. Not good for transaction oriented environment.



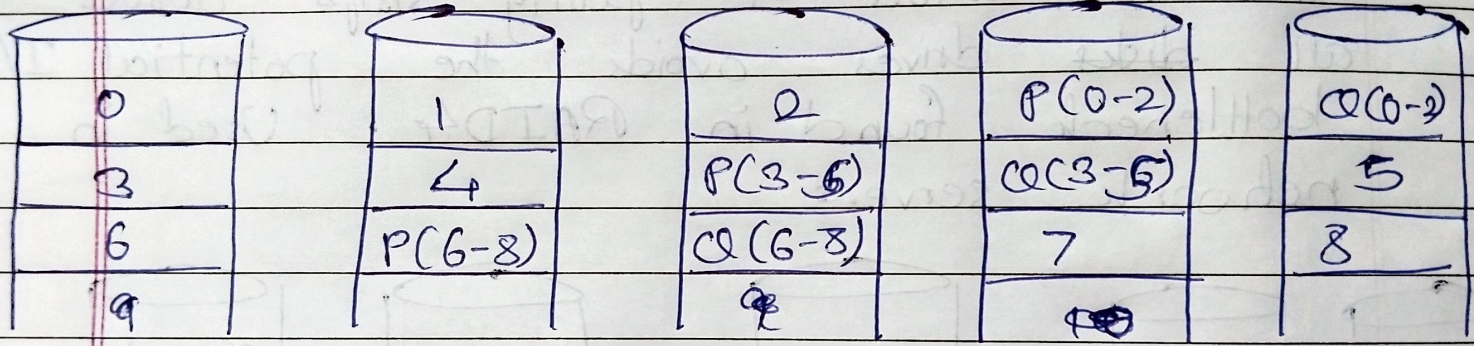
v) ~~Parity~~ RAID 4: Independent access technique is used, in which each member disk operates independently so that I/O requests can be satisfied in parallel, making it suitable for applications that require high I/O ~~requests can be satisfied in parallel~~ rates.
RAIDs 4-6, the strips are very large.
RAID 4, a bit-by-bit parity strip is ~~called~~ calculated across corresponding strips on each data disk, & the parity bits are stored in the corresponding strip on the parity disk.



vi) RAID 5: Similar to RAID 4 but distributes the parity bit across all bits not just one disk. Uses round robin allocation (?). The distribution of parity strips across all ~~disks~~ drives avoids the potential I/O bottleneck found in RAID4. Used in network servers



vii) RAID 6: Two different parity calculations are carried out & stored in separate blocks on different disks. Thus, if RAID 6 requires N arrays then ~~it~~ it consists of N+2 disks. RAID 6 provides extremely high data availability. Three disks would need to failed to cause data to be lost. RAID 6 incurs high ~~penalty~~ write penalty, because each write affects two parity blocks.
Each write operation ~~has~~ requires the disk to read the data, read the first parity, read the second parity, write the data, write the first parity & then finally write the second parity, making a total write penalty of 6.


0
3
6
9
1
4
P(6-8)
2
P(3-5)
Q(6-8)
P(0-2)
Q(3-5)
7
Q(0-2)
5
8