Operating System → Short Note.

- **Intro:**

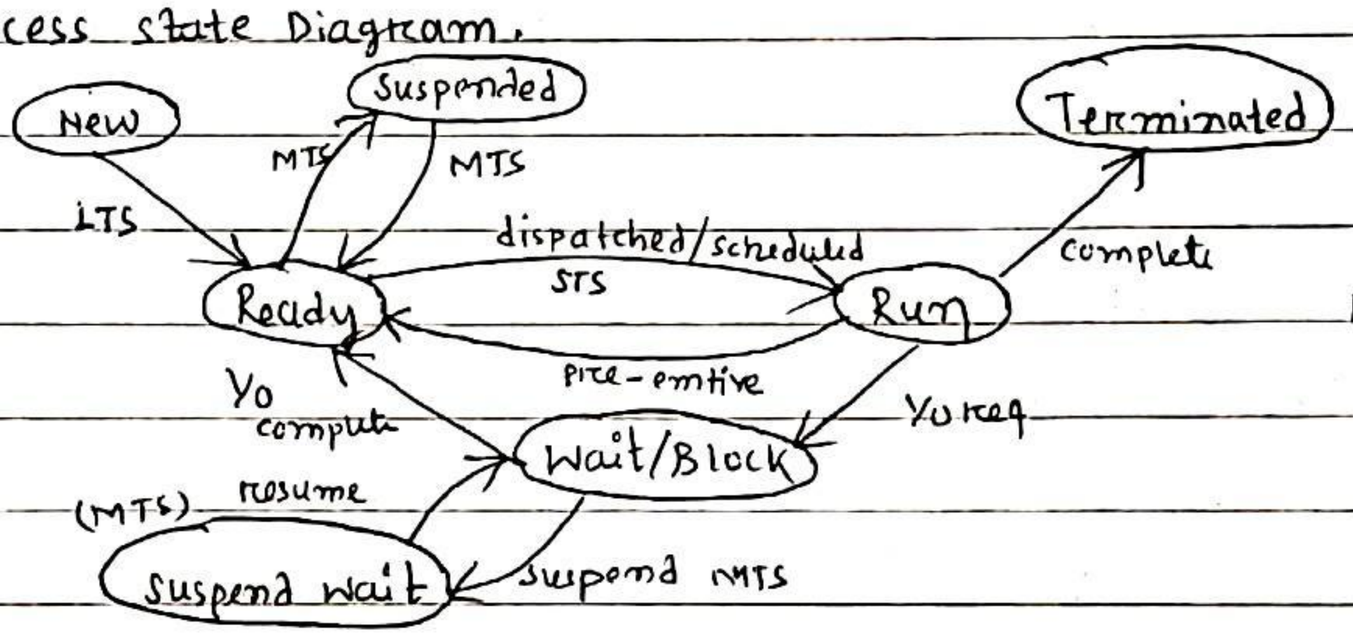
→ Operating system is a software that manage the computer hardware.

- **Process Mannagement:**

- **processes:** process divided in 4 sections (stack, heap, text, data)

→ A process is a program in execution.

→ process state Diagram.



LTS → Long term sche
MTS → Medium term sched
STS → short term sched

→ Min process required for an context switching '2'. Exception → Round Robin's'.

→ FORK: fork() return → -ve, 0, +ve

negative → child creation unsuccessful.

'0' → newly created child.

Positive → Process ID of child process to the parent.

→ If Program contain n fork() calls it will creat $(2^n - 1)$ child pro. **

- **THREADS:**

→ called lightweight process.

→ $CS_T << CS_P$ CS_T → Context switching Time of threat.
CS_P → " " " " process.

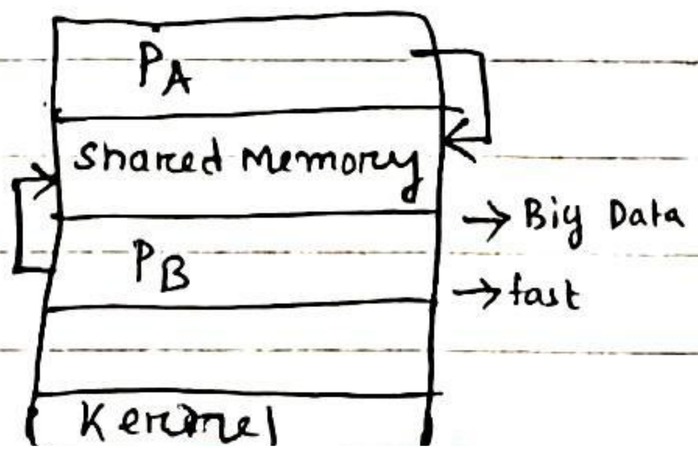
→ Every Thread has it own stack & register.

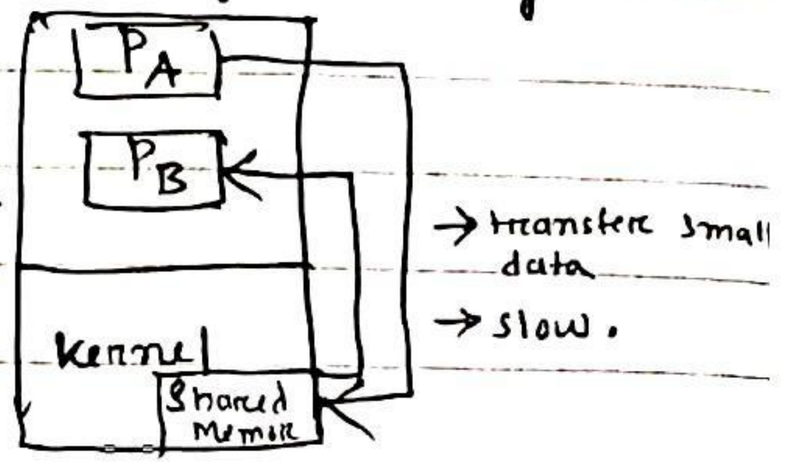
→ Thread → User level thread (Dependent)
→ Kernel Level thread (Independent thread).

→ Thread scheduling also known as GANGR scheduling.

- **INTER-Process Communication (IPC):**

↳ 2 (model) → Shared memory (1) → Message memory (2)





*** Process Synchronization:

→ Synchronization Conditions: point of time.

1. Mutual Exclusion → only '1' process allowed into critical section at any

2. Progress → No process running outside the CS (critical section) should block other interested process from entering into CS when C-S is free.

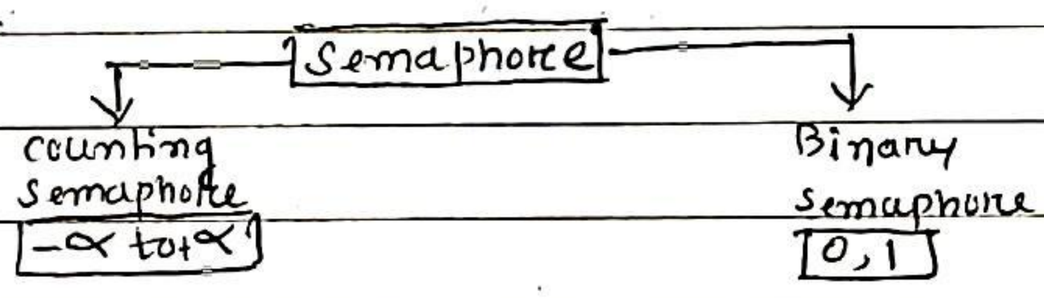
3. Bounded Waiting → No process should not have to wait forever from entering into critical section.

→ Solution type:

Solution	ME	P	Bounded waiting
(I) Lock variable	X	✓	X
(II) Strict Alternation	✓	X	✓
(III) Peterson's Algo	✓	✓	✓
(IV) TSL	✓	✓	X

** Semaphore:

→ is an Int variable used by various process in a mutual exclusive manners to achive synchronization.



→ 2-main operation — (I) Down operation or P() or wait.

(II) Signal operation or V() or release or up()

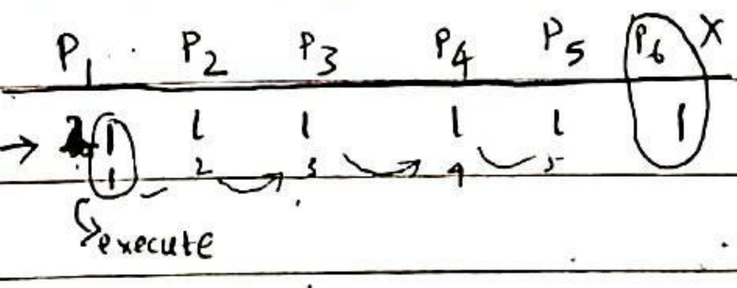
→ The down operation on Binary semaphore is successful only if the semaphore value is '1'.

→ process performing up operation will definatly continue the execution.

P(1) → P(0) | V(0) → V(1).

** DEADLOCK:

→ n process and 6 tapes, if each process req 2 taps to complete execution max value of 'n' is to ensure deadlock free operation "n-1"



***

→ Bankers Algorithms (Max need - Curr Allocated)

process	Max need A B C	current Allocated A B C	Remaining need A B C	current Available A B C
P_0				
P_1				
P_2				

→ Deadlock free sequance ?

***

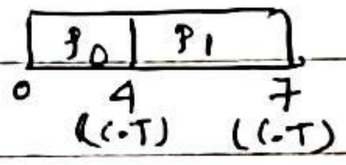
- CPU Scheduling:

→ T.A.T = C.T − A.T | W.T = T.A.T − B.T

(1) FCFS (Fast ruine Fast serve): Criteria → Arrival time, Mode → Non-preemptive

Process	A.T	B.T	C.T	T.A.T	W.T
P_0	0	4	4	4	0
P_1	1	3	7	6	3

Gnuantt chart →



→ In FCFS, if 1st process having largest burst time then it will have drastic effect on avg waiting time of all process. this effect is called CONVOY Effect.

(2) SJF (Shortest Job Fast): (Burst time, Non-preemptive or premptive)
criteria Mode

→ If B.T of process same then schudule process which has lowest Arrival time.

(3) priority Based scheduling: (priority, Non-premptive)

(4) LJF (largest Job First): (Bust, Non-pre or premptive)

(5) Round Robin scheduling: (Time Quantum / time slice), premptive)

Process	A.T	B.T	C.T	T.A.T	W.T
				C.T − A.T	T.A.T − B.T

quantt chart →

Ready Queue → $P_1\ P_2\ P_3$ ....

⇒ Time Quantum (TQ) ↓ → response time decreare.

⇒ TQ ↑ → response time Increare.,

⇒ TQ ↑↑↑ → FCFS

(6) HRRN (Highest Response Ratio Next):

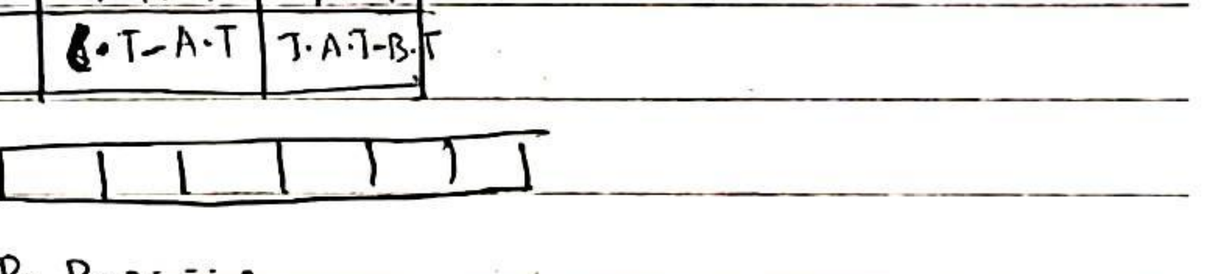
Respone ratio (RR) = $\frac{W+S}{S}$ [W → Waiting time, S → B.T]

(7) CPU & I/O scheduling:

P.NO	A.T	CPU time	I/O time	CPU time

P.NO	A.T	I/O time	CPU time	I/O time

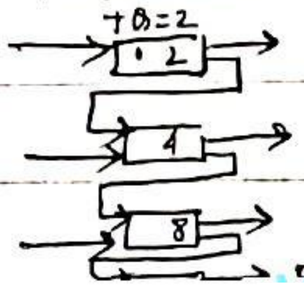
(8) Multi processors scheduling:



→ Guantt chart

(9) Multilevel Queue scheduling: (Suffer from starvation)

*(7) MFQ (Multilevel feedback queue scheduling)



→ avoid starvation.

* Conclusion:

	Starvation		Starvation
1) FCFS	NO	6) LRTF	yes
2) SJF	yes	7) HRRN	yes
3) SRTF	yes	8) priority (NP)	NO
4) RR	NO	9) priority (P)	yes
5) LJF	yes	10) MLQ	yes
		11) MLFQ	NO

*** MEMORY Management:

$2^{10} \rightarrow 1K \mid 2^{20} \rightarrow 1M \mid 2^{30} \rightarrow 1G \mid 2^{40} \rightarrow 1T$

⇒ Memory capacity = no. of word × size of each word.

• RAM chip: no. of ram chip req = $\frac{\text{Memory capacity}}{\text{ram chip size}}$.

→ Memory Management Technique
- Contiguous
  - → fixed partition schemes (Internal fragmentation)
  - → variable partition (External fragmentation)
- Non-contiguous
  - → paging *
  - → multilevel
  - → Inverted
  - → segmentation
  - → segmentation paging

** • partition allocation schema:

→ First: first sufficient partition from top.

→ Best fit: Smallest sufficient parti among free available partition.

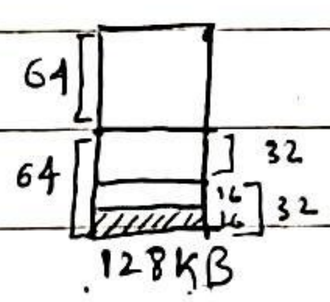
→ Worst fit: Largest sufficient among " " " .

→ Next fit: Work like First fit but it will search for first suff parti from last allocation point.

• Buddy System theory:

→ Initialy memory single continous free block.

→ Whenever req by process come in memory will be divided into 2 half block



**** PAGING (Non-contiguous):

- Logical address or virtual address = n bit
- Logical address space or V.A.S = 2^n Byte.

P.A.S = 2^n B	L.A.S = 2^n B
P.A = n bits	L.A = n bits

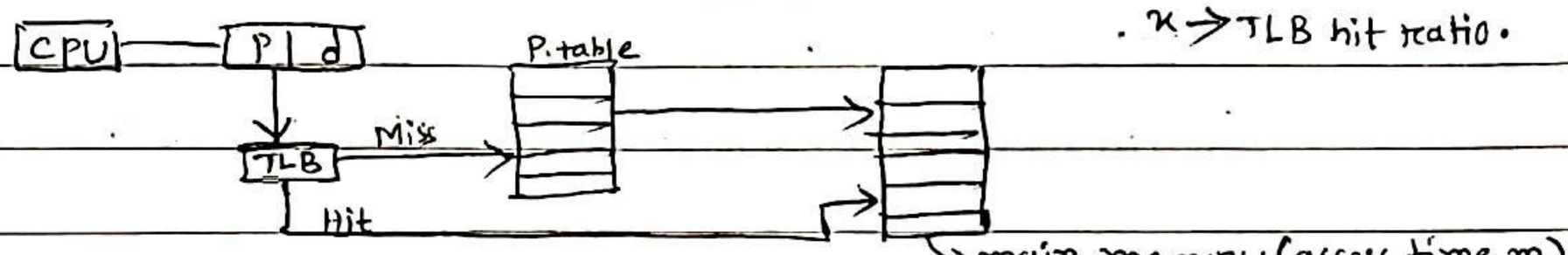
L.A >> P.A

Pages = $\frac{L.A.S}{page\ size}$ # frames = $\frac{P.A.S}{frame\ size}$

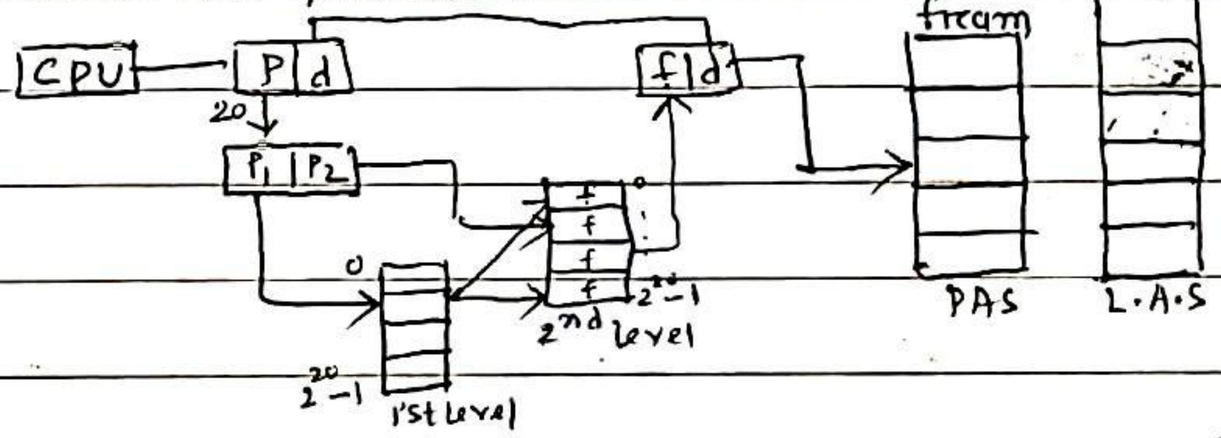
* page size always same as frame size.
* No. of entries in page table is same as no. of pages in L.A.S.
→ Page table entry contain frame no.
→ Paging has Internal fragmentation.
→ If page table stored in the main memory, the formula for effective memory Access time (EMAT) = 2M (M → main memory access time)

**• TLB (Translation Lookaside Buffers):

EMAT = $\kappa(c+m) + (1-\kappa)(c+m+m)$ c → TLB access time, κ → TLB hit ratio.



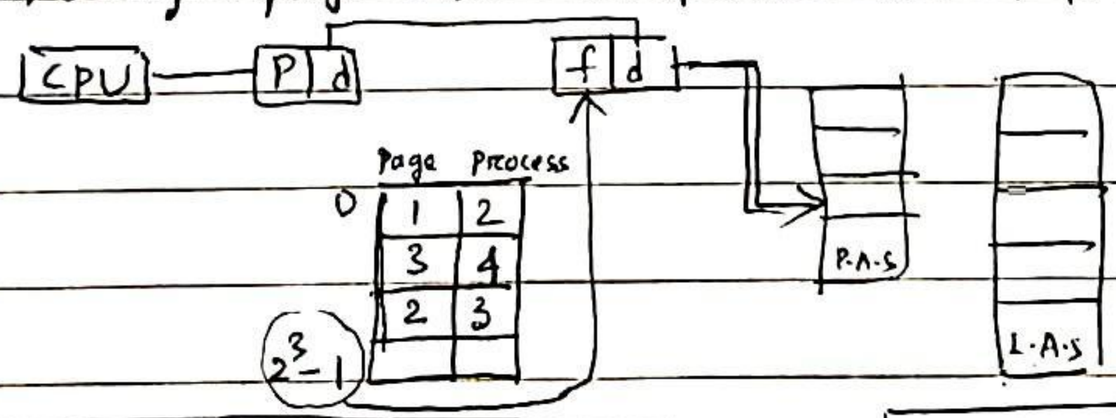
*• MULTI LEVEL PAGING:



→ 2-level paging EMAT = 3m m → main memory access time.
** → 2-level paging with TLB, EMAT = $\kappa(c+m) + (1-\kappa)(c+m+m+m)$

• INVERTED PAGING:

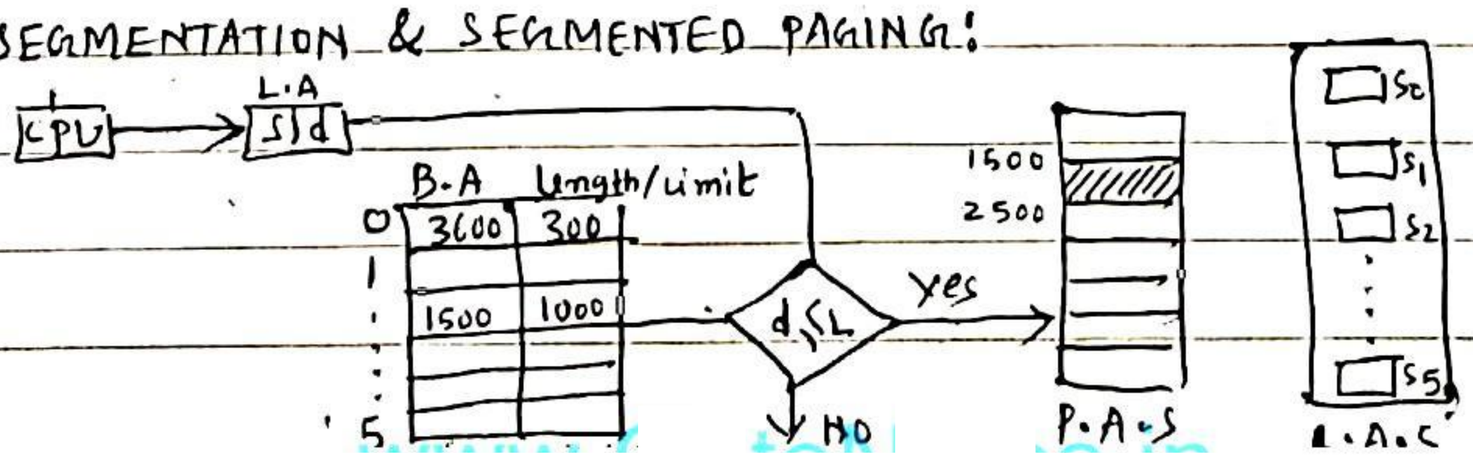
→ only 1 page table maintained for all process.



* → no. of entry = no. of page → no. of entry = no. of frames

→ page Table size = no. of entry * p.T entry size.

• SEGMENTATION & SEGMENTED PAGING:



→ No. of entries in segment table = No. of segment in L.A.S.

→ To achive user view of memory allocation Seg was implemented.

→ Segments will vary in size.

*** • Virtual Memory:

→ V.M implemented using (1) Deman paging (2) Demand Segmentation

-Deman paging $\boxed{EMAT = P * S + (1-P) * M}$

P → page fault rate
S → page fault service time
M → main memory access time

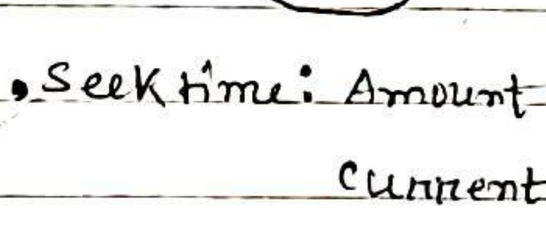
→ $\boxed{1ms = 10^6 ns}$ $\boxed{1sec = 10^9 ns}$

*** • page replacement algorithms:

1) FIFO (First in First out)
2) optimal page replacement
3) LRU (Least recently used)
4) MRU (Most recently used)

→ No. of page fault?

• File and Device Management:



• Seek time: Amount of time taken to move the R/W headers from its current position to Desired track.

• Rotational latency → 1/2 * Rotation time.

• Transfer time, Transfer rate.

* Total time = (seek time + Rotational latency + Transfer time)

$\boxed{1CM = 10MM}$

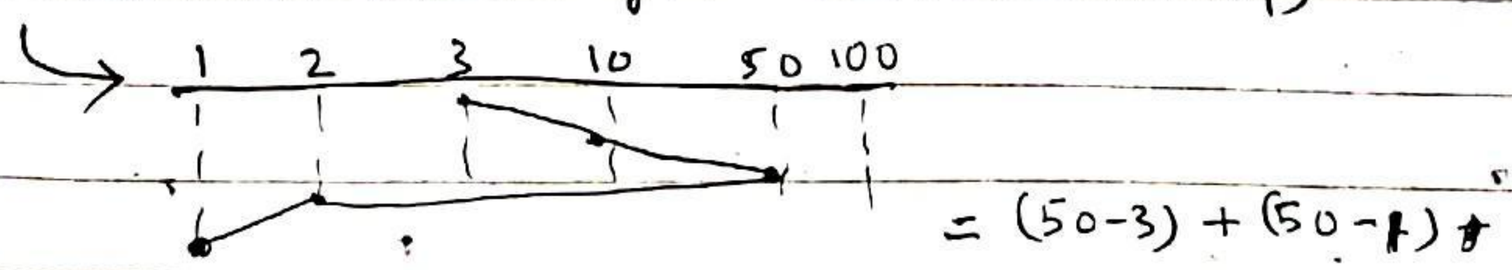
→ No. of tracks = $\frac{\text{Data storage}}{\text{I.T.D}} = \frac{\text{Disk Inner Diameter}}{\text{Inner track Distance}}$

** • DISK SCHEDULING:

(1) FCFS (First Come First Serve)

(3) SSTF (Shortest Seek time first or Nearest track next)

(3) SCAN (R/W header moving toward Right Direction)



(4) C-scan.

(5) LOOK (LOOK for least pending req in the direction of R/W headers)

(6) C-LOOK.