Operating Systems (OS)

1) Basic Introduction → Types, process diggram, System Call
* 2) Process Scheduling → FIFO, SJF, Round Robin etc
3) Process Synchronisation → Semaphore
4) Deadlock & Threads → Banker
* 5) Memory management → Paging, Segmentation, fragmentation, Page Replacement algo
* 6) Disk Scheduling → Scan, CScan, FCFS
7) UNIX Commands
8) File management & Security → Sequential, Random, linked

OS is a system software that interfaces user with the Hardware

User ⟷ Hardware
OS

- agar OS Nhi Hoga toh fir Har command Ke liye Individual programs likhna pdega

- Primary Goal → Provide convinience to the user to access the Hardware
- Throughput → No. of tasks per unit time (Linux Ka zyada)
- Resource Management - Kisi Resource ko multiple user access kr Rhe Hai toh Kitna Kisko dena Hai yeh Bhi OS Krta Hai

- Process management → Multiple process ko execute krne ka management (CPU Scheduling)
- Storage management → Hard disk (file system) (No constraint of size)
- Memory " → RAM (Memory limited as Primary memory so swapping of data)

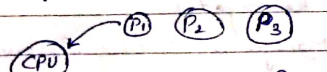
- Security & Privacy → Password to login into windows, processes ke Bich me Bhi Security so that data Interfere na kre

User → Applicat^n → OS → Hardware (CPU, RAM etc)
↓
work through System Call
(without Application Hum Terminal se Kaam Kr Skte Hai)

→ Types of Operating System

1) Batch OS : Punch Cards, Paper Tape, Mag. Tape → operator (offline tapes ka Result Bhane ke liye)
↓
Similar work ke Batches
B_1 B_2 B_3
↓
CPU ⟲ I/O
Ek Bar me Ek hi Job ho Rhi so idle Time Jyada ← Idle

* 2) Multiprogrammed OS — Non Preemptive (RAM)



Idle nahi kam ho Jayegi

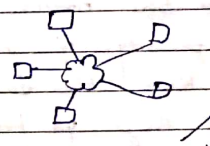
Now agar CPU ko P1 Process mila Hai toh Phle voh Pura P1 Krega Fir P_2 pe Jayega But yes agar P_1 ko Kbi Aur Jana Pda eg I/O toh P_2 pe CPU Jayega Nhi toh Jab tak P_1 CPU ke pas Hai CPU Nhi utha skta

3) Multitasking / Time sharing OS → Preemptive

CPU ne time allot krdiya i.e ki agar ek Process ek particular Time me Hogya toh theek vrna voh Dusre process pe Chala jayega & usse baad ke liye Rkh dega.
Emphasis → Response Time is less (Idle yha pe Bhi Bohot Kam)

4) Real Time OS
Work ek dum Batch Jaisa Hi Hai but bass Time pe Bohot Emphasis Hai ki Time Bohot Kam Ho
Types → Hard – Mtlb very hard emphasis on time eg- missile
→ Soft – Utna Bhi Hard constraint nhi Hai time ka eg- youtube live

5) Distributed OS → Multiple machines (loosely coupled), (Geographically alag alag)



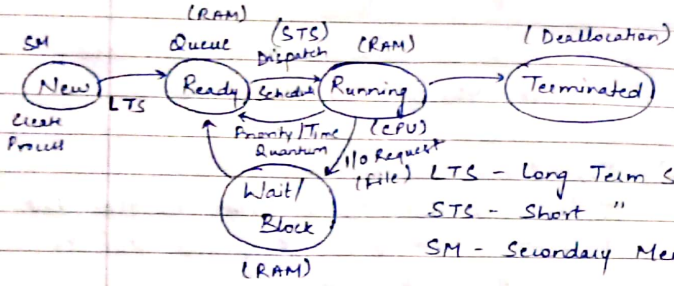
Availability, Fault tolerance
Scalability (aj Computation Power is x kal 2x toh easily ho Jayega)

6) Clustered OS → Multiple machines but locally located ek hi Network me

(Ek Machine ki Tarah kaam kr Rha Hai)

7) Embedded OS → Jo Ek Fixed Functionality pe hi kaam Kr Rha Hai
eg- Microwave → Heat Krta Hai toh bas Vhi Krega

→ Process States

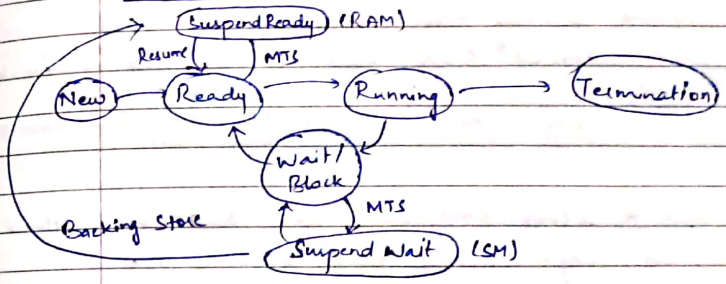


LTS - Long Term Scheduler
STS - Short " "
SM - Secondary Memory

Initially thum SM me kisi process ko create Krte Hai but as RAM thmare pas limited Hai so LTS would decide ki kisko SM se age Bheja Hai (Ready Queue) me now depending of no. of CPUs utne no. of process Running state me chale Jayeg but we assume uni processing so single process hi Jayega ye kaam Krega STS but abhi bhi process is in RAM. Now, According to the priority if VIP Process comes toh phle voh Run Ho Jayegi & current vali Baad me Hogi ya Fir On the Basis of Time Quantum esa Hoga like Fixed allotted Time for Multitasking. Now lets suppose Ek Process ko Running state me I/O Request Krni pd Gyi i.e File Requirement Hai From the Secondary memory so CPU us process ko (wait) State of RAM me daaldega & Baaki Processes krta Rhega Jaise hi File aa Jayegi toh CPU us Process ko First Ready state me daaldega taaki Future me uska no. aa Jaye

Some additional States

Agar (wait) Block Bhar Jaye toh Kuchh der ke liye Jab tak khali Nhi Ho Jata thum use (Suspend wait) State which is on Secondary memory me Rkh dete Hai Similarly if Ready Queue Bhar Jaye & Koi VIP Process aye toh thodi der (Suspend Ready) State me process ko Rkh dete Hai yeh vala kaam Krta Hai Medium Term Scheduler (MTS)



Aur agar Bilkul hi khali Nhi Ho Rha Suspend wait toh direct Suspend Ready me le Jaate Hai Process ko i.e called Backing Store

→ Some UNIX commands

chmod → change mode
3 Things

u → r (read) → 4
g → w (write) → 2
o → x (execute) → 1
↳ octal

r w x | w x | r
u | g | o

if we write (-) than permission vapis lena & " (+) " " dena

∴ chmod go+r note (g & o Ko read permission di)
chmod ugo=r note (u, g, o Ko read " ")
chmod u+r, g+r, o-x note (u Ko read & (g) Ko read di but (o) u execute leli vapis)

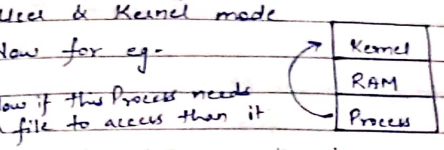
→ writing these commands in octal
chmod 555 note (u, g, o Ko 4+1=5 i.e. read, execute ki Permission dedi)
similarly 666 " (4+2=6 → Read, write ki permission)

* lseek → will take Read/write Head
eg- 1 2 3 4 5 6 7 8 9 0
By default R/W Head at Index 0
lseek (n, 10, SEEK_CUR)
↳ n file me current position se 10 age le Jao i.e Index 10

lseek (n, 5, SEEK_SET) ; n is file descriptor
↓
Mujhe Index 5 Pe le Jake Head set Krdo
SEEK_END Jab use Hota Hai toh no. -ve likhte Hai

→ System Call

2 modes of operation are there that are User & Kernel mode
Now for eg-



Now if the Process needs a file to access than it doesn't have the authority it will ask the Kernel (OS) & than that will do System calls & give the files
eg- printf is system call to print on screen

System Call
↳ File Related : open(), read(), write(), close(), Createfile etc
↳ Device Related : Read, write, Reposition, ioctl (future control), fcntl,
↳ Information : getPid (Process id), attributes, get System time & data
↳ Process control : Load, Execute, abort, fork, Wait, signal, allocate etc
↳ Communication : Pipe(), Creat/delete Connections, Shmget()

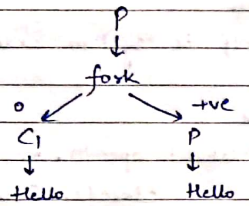
→ Fork() (System Call)
To create a child process i.e ek dum ek clone create Ho Jayega Jiski apni ek id Hogi & process parallely Hoga

Child Hmesha 0 ... Parent Hmesha 1

Page No.
Date

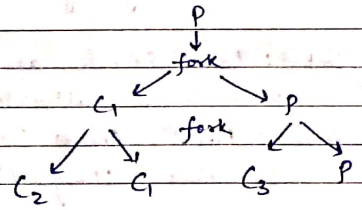
fork() → 0 (+ve) Child Process ki id
→ +1 Parent
→ -1 → Child nahi" created } To not be considered

eg- (i)
```
main() {
    fork();
    printf("Hello");
}
```



2 Times Hello Print

(ii)
```
fork();
fork();
printf("Hello");
```

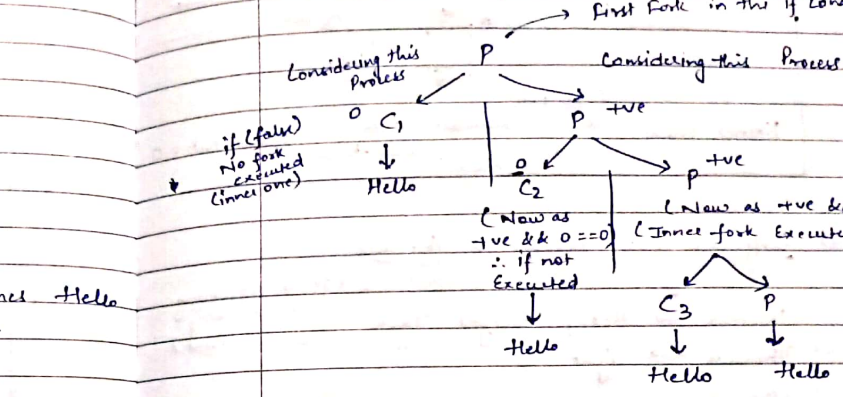


4 Times Hello Print

4 Executions & 3 Child Processes

∴ 2^n → No. of Executions
$2^n - 1$ → " " Child Processes
n = no. of times fork called

eg-
```
if (fork() && fork())
    fork();
    printf("Hello")
    return 0;
```



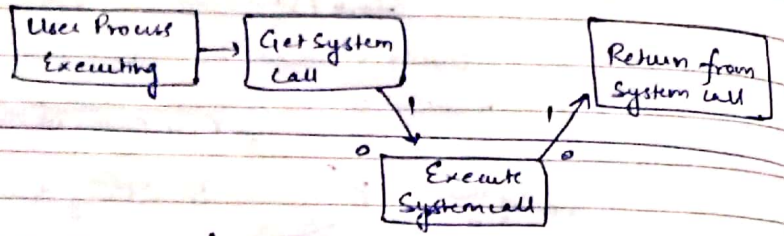
∴ 4 Times Hello Print

Similar Quest" with (||) OR operator

→ User Mode v/s Kernel Mode

Jab bhi hum kai App use krte hai than hm use in User mode but app internally Jo kaam krta hai i.e in Kernel mode
eg - Drivers saare But (µP) switch krta Rhta hai
eg-- code likha toh user mode execute in Kernel mode

User mode — Mode Bit = 1

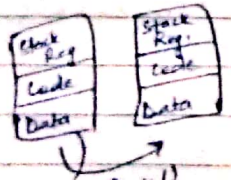


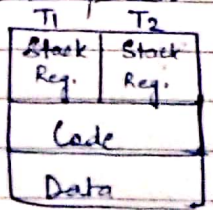
Kernel mode (as files are in Hardware & to access them we need to be in Kernel mode) — Mode Bit = 0

→ Difference b/w Process & Threads (Multiprocessing OR Multitasking Environment)

Process → Heavy weight Task
Threads → Light " "

Process	Threads
1) System Calls involved in Process	There is no system call Involved (user level)
2) OS Treats different processes differently (different PID for Processes)	All user level threads treated as single task for OS
3) Different process have diff. copies of data, files, code	Threads share same copy of code & data
4) Context Switching is Slower	Faster (Time Sliring Environment)
5) Blocking a process will not block another	Blocking a Thread will block entire process
6) Independent	Interdependent





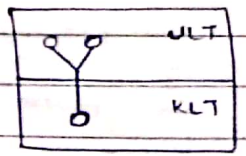
(Memory Share toh Fast)

* Process me EK Child Process Generate ho Jaata hai i.e EK Pura Clone generate ho Jaata Hai (Fork System call)
* Whereas thread me same Code & data share Hota Hai but usme multiple threads Bn Jayenge

→ User level thread	Kernel level thread
1) Managed by user level library	Managed by OS (System calls)
2) Typically Fast	Relatively Slower Slower (because OS Ka Role Hai Bar Bar System call invoke)
3) Context Switching is Fast	
4) If one user level thread perform blocking opr^{n} the entire process gets Blocked	If one Kernel level thread blocked, No affect on others

Process > KLT > ULT
(CT)
↳ Context Switching EK se Dusre pr shifting ka Time

(Hybrid model)

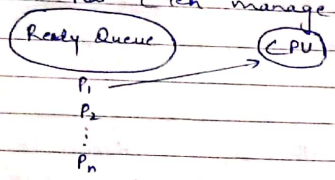


Mapping is Many to one, Many to many, one to one

* either ULT or KLT Both are threads so both of them share Code & Data

→ **Scheduling Algorithms (SA)**

Way ~~to~~ Hai Ki Kis process ko Ready Queue se uthake CPU ko processing ke liye dena Hai (Yeh manage krta Hai)



SA

Pre-emptive

*
- SRTF (Shortest Remaining Time First)
- LRTF (Longest Remaining Time First)

*
- Round Robin
- Priority Based (It can be a part of Non preemptive also)

(But idhar ho skta Hai Ki Ek Running process ko dobara bhi Ready Queue me dalde)

Non-Pre-emptive

- FCFS (First Come first Serve)
- SJF (Shortest Job First)
- LJF (Longest Job First)
- HRRN (Highest Response Ratio next)
- Multiple Queue

(Non preemptive means agar ek process processing ke liye gya toh pura hoga fir second process)

→ **Times at CPU Scheduling**

1) Arrival Time - Time at which Process enter the Ready Queue or State (Point of time)

2) Burst Time - Time Reqd. by a process to get execute on CPU (It is a duration)

3) Completion Time - Time at which process completes it's execution (Point of time)

4) Turn Around Time = (Completion - Arrival)

5) Waiting Time = (Turn Around - Burst time)

6) Response Time = (Point of Time)
[(The time at which a process get CPU first time) - (Arrival Time)]

Now, 2 Types of Bounds are there

- CPU Bound → Mtlb process ka Execution ka Time CPU pe Kitna
- I/o Bound → Abh pure process me i/o bhi lene pd skte Hai toh usme bhi time lgta Hai so that is waiting time

* Response ka mtlb Hai Kab uspe Respond Kiya gya.

→ **FCFS** → Jo phle ayega usko Serve Krenge

* Non-Preemptive me response time ka normally Koi use nhi as same to same waiting time jitna hi ayega.
* Gantt Chart me jo left Subscript pe time likha hota Hai i.e the First time when process got the CPU
* SJF me Criteria is that Jis process ka Burst time kam hoga hum usse sbse phle Run krvayenge
(Tie Breaker is arrival time Ki Jo phle aya usse phle agar same burst time & if vo bhi same toh fir PID Jiski kam usse phle)

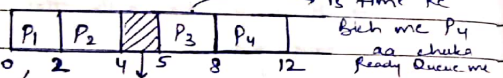
* SRTF is SJF with mode = preemptive

Q) FCFS

Process No.	Arrival Time	Burst	Completion	TAT	WT	RT
P_1	0	2	2	2	0	0
P_2	1	2	4	3	1	1
P_3	5	3	8	3	0	0
P_4	6	4	12	6	2	2

Criteria = "Arrival Time"
Mode = "Non Preemptive"
(Jo pehle ayega voh pura process hoga fir aage jayega)

Gantt Chart



is time ke bich me P_4 aa chuka Ready Queue me

Is time ke Ready Queue me bich P_2 arrive ho chuka hai

Idle → Time

Bcoz Koi Bhi Process Ready Queue me nhi hai

Ready Queue ~~P_1~~ ~~P_2~~ ~~P_3~~ ~~P_4~~

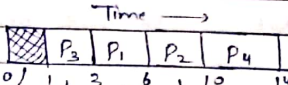
Avg. TAT = $\frac{14}{4}$; Avg. WT = $\frac{3}{4}$

Q) SJF → Criteria = "Burst time" (Jiska chhota usko pehle)
Mode = Non Preemptive

Process	Arrival	Burst	Completion	TAT	WT	RT
P_1	1	3	6	5	2	2
P_2	2	4	10	8	4	4
P_3	1	2	3	2	0	0
P_4	4	4	14	10	6	6

Time →

Gantt Chart



idle as No arriving at 0

P_1, P_3 in Ready Queue But Selection acc. to small Burst time (isi Bich P_2 Bhi arrive kr gya)

Out of P_1, P_2 (isi Bich P_4 Ready Queue me arrive kr gya)

out of P_2 P_4 as done ko Burst time same to acc. to arrival time P_2 select kiya as pehle aya

Ready Queue P_1 | P_3 | P_2 | P_4

* Criteria Kewal tab use Krna Hai when arrival hi jyada ke ho abh maano agar arrival hi ek ka ho to there is no competition to vhi process hoga

Q) SRTF = SJF with preemption

* Round Robin → Time Quantum (preemptive)
* Context Switching me partially done kaam ko save kro PCB (Process control Block) me & vapis Ready Queue me le Jao
* Round Robin me CPU ko Bilkul Bhi idle nhi Krne ya to process complete hogi ya Time Quantum Jitna process hogi
* WT ke liye Burst time ek dum original vala use Krte hai Jo initially tha naaki Jo update Krke aya
* No. of Context Switching = Gantt Chart me first aur last line ko exclude Krke Baaki ki line calculate krlo (No. of times Jab Running process ko save kiya & Dusre process ko laye)

Q) SRTF → SJF with preemptive

Process No.	Arrival	Burst	Completion	TAT	WT	RT
P_1	0	~~6~~ ~~5~~ 3	9	9	4	0
~~P_2~~	1	~~3~~ ~~2~~ ~~1~~ 0	4	3	0	0
P_3	2	~~4~~ 0	13	11	7	7
P_4	4	~~1~~ 0	5	1	0	0

Criteria = Burst Time Mode = Preemptive

Preemptive me agar Time Quantum is not given then the process ko 1 unit time ke liye chalayega & constantly will check the Ready Queue for Shortest Burst time process (updated one) & usse execute krega (Running se vapas Ready Queue bhejta rhega)

Gantt Chart

P_1	P_2	P_2	P_2	P_4	P_1	P_1	P_1	P_1	P_3
0	1	2	3	4	5	6	7	8	9	13

Ready Queue

P_1P_2 → t=1

$P_1P_2P_3$ → t=2

" → t=3

$P_1P_3P_4$ → t=4

P_1P_3 → t=5 (Har 1 unit time ke baad checking chalti rhegi) (& if Burst time same toh arrival time)

at t=0 No competition so ek jo hai voh process ho jayega

Now for WT = TAT − BT

Now this should be the original one that was provided Initially & not to updated one

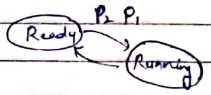
Criteria = Time Quantum

Mode = Preemptive

Q) Round Robin (TQ) → Preemptive (But here Time Quantum is given) so will be processing each process only for that time or if the process Burst time gets over before that

Process	Arrival	Burst	Complet"	TAT	WT	RT
P_1	0	~~8~~ ~~3~~ ~~1~~ 0	12	12	7	0
P_2	1	~~4~~ ~~2~~ 0	11	10	6	1
P_3	2	~~2~~ 0	6	4	2	2
P_4	4	~~1~~ 0	9	5	4	4

Given TQ = 2



Ready Queue

~~P_1~~	~~P_2~~	~~P_3~~	~~P_1~~	~~P_4~~	P_2	P_1
0			t=2			

Running (Gantt Chart)

P_1	P_2	P_3	P_1	P_4	P_2	P_1
0	2	4	6	8	9	11	12

→ Time

↓ Zaruri nhi ki TQ Tak hi chale 1 unit me Burst time khtm to usse hi chalega

No. of Context Switching = Gantt Chart me no. of Bich Ki lines = 6 (First & last Exclude Krdo)

Har process ko TQ ke liye chalake vapis Ready Queue me push krta jayega (Context Switching) & fir Ready Queue vali ko chalayega

* Har case me Tie Breaker pe use Arrival Time i.e First Come First Serve & if that also same than Pid that is ki list format/Sequence me de Rkha hai usme process krdo i.e P_1 P_2 me se (P_1) Ko Pehle

* Idle Ratio = Idle Time / Total ; Usage Time = 1 - Idle Ratio

* Multilevel Queue Scheduling means ki alag alag Types of processes ke liye alag Ready Queues to use & also har different Queue pe alag se scheduling algo lgao now that can be either same or different

* Multilevel Feedback Queue is used to save the lower priority process from starvation i.e agar low " " Ko agar process Hona hai toh pehle High priority Hongi Fir uska no. ayega but Suppose agar High priority Bohot Jyada hai toh uska Bohot der Baad no. ayega so this is starvation so in this we eventually with specified time Quantum upgrade the priority of lower process to save from starvation

Q) Priority Scheduling

Criteria = "Priority" Mode = Preemptive

↓

Priority No. given Hoga & Yeh Bhi Btaya Hoga that higher the No. higher the priority or lower the no. higher the Priority

Process	Priority	Arrival	Burst	Complet"	TAT	WT
P_1	10	0	5,4,0	12	12	7
P_2	20	1	4,3,0	8	7	3
~~P_3~~	30	2	2,1,0	4	2	0
~~P_4~~	40	4	1,0	5	1	0

[Higher the priority no., higher the priority]

(As preemptive so kuch Chala & than High Priority vala Process Hoga)

Gantt Chart

P_1	P_2	P_3	P_3	P_4	P_2	P_1
0	1	2	3	4	5	8	12

Ready Queue

$P_1 \rightarrow t=0$ $P_1P_2 \rightarrow t=1$ $P_3P_1P_2 \rightarrow t=2$

$P_1P_2P_3 \rightarrow t=3$ $P_1P_3P_4 \rightarrow t=4$

$P_1P_2 \rightarrow t=5$

Q) This Question is mainly to show variation in Burst time

Sometimes Burst time is given as Bifurcation of 3 Times → CPU - I/O - CPU

↓ Initially Jab aya process

↓ I/O (In this part" CPU par nhi Hoti Process & us particular time period Jab Tak uska I/O Complete nhi Ho Jata CPU par aa Bhi Nhi Skti)

↓ Again Final Processing

Criteria = "Priority" ; Mode = Preemptive

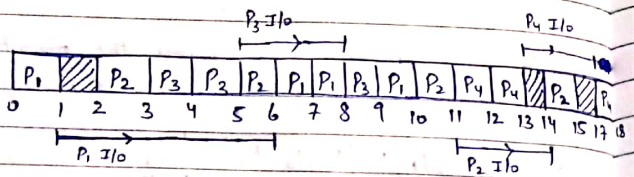
(Low No High Priority)

Page No.
Date

Burst Time

Process	AT	Priority	CPU	I/O	CPU	CT
~~P_1~~	0	2	~~X~~o	~~8~~o	~~8~~X~~X~~o	10
~~P_2~~	2	3	~~8~~X~~X~~o	~~8~~o	~~X~~o	15
~~P_3~~	3	1	~~X~~X~~X~~o	~~8~~o	~~X~~o	9
P_4	3	4	~~X~~X~~X~~o	~~X~~o	1	18

Gannt chart



Ready Q

$P_1 \rightarrow t=0$ ~~P_2~~ $\rightarrow t=1$ (Gone to do I/o) P_1 ~~P_2~~ $\rightarrow t=2$

$P_2 P_3 P_4 \rightarrow t=3$ $P_2 P_3 P_4 \rightarrow t=4$ $P_2 P_4 \rightarrow t=5$

$P_1 P_2 P_4 \rightarrow t=6$ $P_1 P_2 P_4 \rightarrow t=7$ $P_1 P_2 P_3 P_4 \rightarrow t=8$

$P_1 P_2 P_4 \rightarrow t=9$ $P_2 P_4 \rightarrow t=10$ $P_4 \rightarrow t=11, 12,$

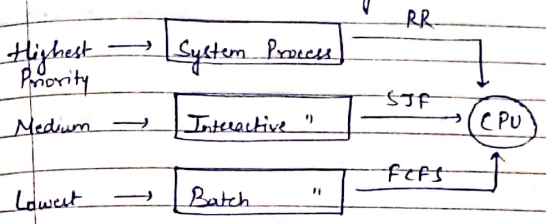
$\rightarrow t=13$ $P_2 \rightarrow t=14$ $\rightarrow t=15, 16, 17$ $P_4 \rightarrow t=18$

Idle Ratio = $\frac{\text{Idle Time (in G. Chart)}}{\text{Total Time}} = \frac{4}{18}$

Usage Ratio = 1 − Idle Ratio = $\frac{14}{18}$

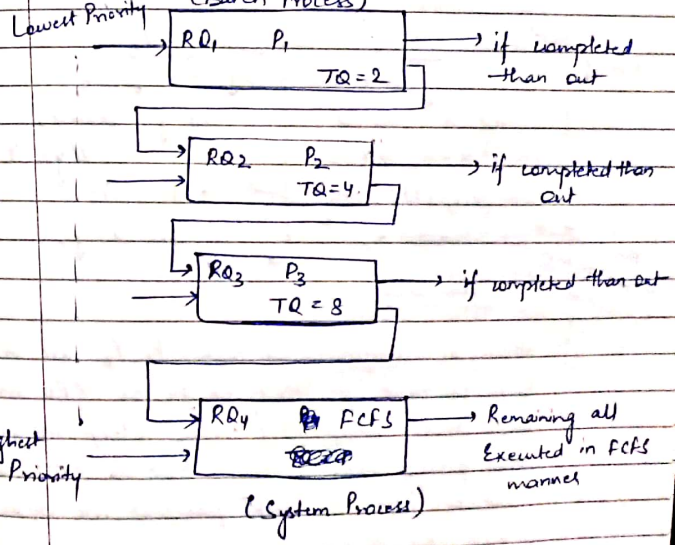
Page No.
Date

→ Multilevel Queue Scheduling



System Process is like an Interrupt
∴ High Priority kbtm thagi thbi low vali ka number ayega could lead to Starvation so solution is as follows :-

→ Multilevel Feedback Queue (Batch Process)



eg - P_1 = ~~14~~ 17 15 (Batch Process)

$\not\in 0$

→ Process Synchronizat"

Processes can be series or Parallel (multiprocessing)

Processes
- Cooperative (Ek ki Execut" dusre pe Effect daalta Hai) (i.e they share something)
- Independent (No Effect on each other)

Share
- Variable
- memory
- Code
- Resources ← CPU, Hardware

Hum yha Cooperative ki hi Baat Krenge

eg- 2 Processes in parallel

Racing Cond"

int shared = ~~6~~ 4

P_1	Context Switching	P_2
s		s
int x = shared		int y = shared;
x++; //x=6		y--; //y=4
sleep(1);		sleep(1);
shared = x;		shared = y;

This command is use to implement Preemption

Terminated

Here as uniprocessor toh P_1, P_2 are in !! But Ek time pe kisi ek hi ko CPU mil skta so, here sleep(1)// means sleep or pause that process for 1 sec But as CPU will not be idle so voh parallel me dusre process ko dhundega

Now, as ek Bar (++) hua & ek bar (--) hua so shared = 5 Hona Chahiye But yha (4) ya (6) [illegible] & creates problems

→ Producer Consumer Problem

Code written for Producer (producing products) & Consumer (consuming products) from the common shared Buffer

n = 8

Buffer [0...n-1]

0	~~x1~~
1	x2
2	x3
3	x4
4	
5	
6	
7	

out = ~~0~~ 1 | in = ~~8~~ 4 | count = ~~8~~ 4 2 | Rp = ~~8~~ 4

```
void consumer (void)
{ int itemC;
  while (true)                    // Buffer Empty so ∞ loop
  { while (count==0);
    itemC = Buffer (out);
    out = (out+1) mod n;          // Rc = ~~8~~ 2
    count = count - 1;
    process_item (itemC);
  }
}
int count = 0;
```

Internally Implemented as

```
load Rc, m[count]   I1
DECR Rc             I2
Store m[count], Rc  I3
```

Produces ↓

```
Void Producer (void)
{ int itemP;
  while (true){
    Produce item (itemP)
    while (count==n);             // Buffer full so ∞ loop
    Buffer [in] = itemP;
    in = (in+1) mod n;
    count = count + 1;
  }
}
```

```
I1  load [Rp], m[count]
I2  INCR Rp
I3  Store m[count], Rp
```

(Internally Implemented)

in & out are like Iterators

Case-I : Synchronised way se done kr Rhe the toh koi issue nhi ata

Case-II : → (idhar process preempt hua context switching hoggyi)

Producer I_1 I_2 Consumer I_1 I_2 Producer I_3

→ Series of Invoking Instructⁿs

Is Series me Jab Execut" Hoti Hai toh process Synchronised nhi Hoti & due to that answer of count comes out to be Wrong i.e count = 2 dikha Rha asal me ana Chahiye 3

→ Printer - Spooler problem (loss of data)

as 1 printer But processes are many & as peripheral device so slow abh ek Spooler Bna dete Hai Jisme print Krne vale documents add Krdete Hai

Spooler Directory

0	f1.Doc
1	f2.Doc
2	f3.DOC
3	
4	

→ Printer

I 1) load R_i, m[in]
" 2) Store CD[R_i], "F-N" (f1.doc)
" 3) INCR R_i
" 4) Store m[in], R_i

(Process) IN [0]
P_i f1.doc
R_i [0]

IN = 3 . Now lets suppose 3 Entries toh directory me "phle se hi hai Now we want ki 2 aur dal Jaye toh f4.doc & f5.doc ke liye set of code chalayenge

P_1 → f4.doc
P_2 → f5.doc
→ Parallel Processes
Now as cooperated Process

∴ P_1 I_1 I_2 I_3 | P_2 I_1 I_2 I_3 | I_4 | I_4
→ preempt Hogya process & P_2 process Hona shuru Hogya Now is series ki Execut" se

0	f1.doc
1	f2.doc
2	f3.doc
3	f4.doc / f5.doc

IN [3] X 4

R_1 R_2
3 4 3 4

∴ As Non Synchronised so loss of data occurs.

* Entry Section ka code esa Ho ki agar ek Process entry section ke code ko pass kr gya toh dusra process use pass na kr Paye taaki synchronise Ho Jaye & Racing Cond" na aye
* First 2 are primary Rules last 2 are secondary
* if P_1 & P_2 has entered Critical Sect" than P_2 cannot Enter (Mutual Exclusion) (Koi Ek hi ek Time pe enter kr Paa Rha Hai)
* Esi koi Restrict" Nhi Honi Chahiye ki Ek Type ke Hardware pe chale & dusre pe nhi Sol" must be universal Har Type ke OS ya Hardware pe Chalna Chahiye

→ Synchronisation Techniques

Critical Part - It is part of a program where shared Resources are accessed by various processes (Cooperative)

P_1 | P_2

#include —
main()

Non Critical Part

This part includes independent variables jo dono processes ke alag alag hai nothing in common

Entry Section

Critical Section
count++

Exit Sect^n

Entry Section

Critical Sect^n
count--

Exit Sect^n

This part includes common variables used by both the processes eg – count

Now for synchronising the process these Entry Sect^n & Exit sect^n are made
These are set of code i.e entry sect^n ka set of code agar ek process qualify krega tbhi Critical sect^n (CS) me ghus skta hai
Nhi toh nhi.

Now,
Synchronical^n Mechanism
4 Conditions

Primary Cond^n (Mandatory)
i) Mutual Exclusion
ii) Progress
iii) Bounded Wait

Secondary
iv) No Assumption Related to Hardware & Speed

i) Mutual Exclusion – Ki Agar ek Process Entry Sect^n pass krke CS me Jaa chuka hai toh Dusra process nhi Jaa skta ek Bar me kewal ek hi Jaa Skta hai

P_1 P_2
CS
P_1 P_2

ii) Progress – Kbhi Kbhi esa ho skta hai ki (P_1) Jaana chah Rha hai CS me But (P_2) ke Entry Sect^n me kuchh esa likha hai ki voh (P_1) ko Bhi Rok Rha hai So that Shouldn't happen

So esa nhi Hona Chahiye ki if ek process Jana chah Rha hai toh Dusra use Roke esa code likhe

P_1 P_2
CS

Hone chahiye ki dono ek dusre ko progress Kone de

iii) Bounded wait

Esa Na ho ki Ek process CS part ko khtm krta hai & ek dum se firse vha chala Jaata hai this would lead to Starvat^n for P_2

CS

P_1	P_2	
1	0	
2	0	10 – 1 ✓ (Yeh Chalta hai)
3	0	
⋮		
∞		But ∞ – 0 / or ∞ – 1 ✗ (Yeh nhi Hone chahiye)

iv) No Assumpt^n → MHb code for the solution must be universal Hai type / speed ke Hardware pe chalna chahiye Kisi specific pe nhi

→ Semaphore – It is a way or tool to prevent Racing Condition in process Synchronisation

Semaphore is an integer variable which is used in mutual Exclusive manner by various concurrent cooperative processes in order to achieve synchronisation.

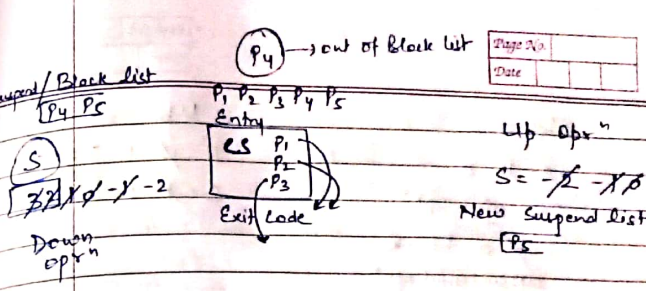
Semaphore
- Counting ($-\infty$ to $+\infty$)
- Binary (0, 1) → value

Name of opr^n:
- P(), Down, Wait ← Entry Sect^n
- Critical Section
- V(), Up, Signal, ← Exit Sect^n (post, Release)

```
Down (Semaphore S)
{
  S.value = S.value - 1;
  if (S.value < 0)
  {
    put Process (PCB) in
    Suspended list
    Sleep()
  }
  else
    return
}
```

```
Up (Semaphore S)
{
  S.value = S.value + 1;
  if (S.value <= 0)
  {
    select a Process
    from suspended list
    wake up()
  }
}
```

wake up() → This will bring the process from Block list to Ready Queue



if (S) value is (n) than (n) no. of Processes can enter the CS

eg ⇒ S = 10 → 10 Process can enter & if S = -2 than it means 2 " in Block list

Q) S = 10
Operat^{n}s performed = 6P, 4V
find new S ?
S = 10 − 6P + 4V (6 Times Decrement, 4 " Increment)
= 10 − 6 + 4 = 8

Q) S = 17 ; opr^{n}s 5P, 3V, 1P
S' = 17 − 5 + 3 − 1 = 14

→ Binary Semaphore
- 0
- 1

```
Down (Semaphore S)
{
  if (S.value == 1)
  { S.value = 0; }
  else
  { Block this process
    & place in
    Suspend list
    sleep()
  }
}
```

```
Up (Semaphore S)
{ if (Suspend list is Empty)
  { S.value = 1; }
  else
  { select a Process
    from suspend list
    & wake up()
  }
}
```

Page No.
Date

Down

S = ✗ 0
Successful opr^n
S = ∅ 0 Block
Unsuccessful opr^n

Up

S = ∅ 1 [if Suspend List Empty]
S = ✗ 1

eg -

Block
P1

Ready Q
(P1)

S = ✗ 0

(if we take S = 0 Both process would go in Block list)

P1
Down (S)
CS
Up (S)

P2
Down (S)
CS P2
Up (S)

No need to Remember the Code Just (S) ke oprn's dekhlo

Q) Each process P_i {i = 1 to 9} execute the following Code

```
Repeat
P(mutex)
CS
V(mutex)
Forever
```

Process P10 Execute the following Code

```
Repeat
V(mutex) → Entry
CS
V(mutex) → Exit
Forever
```

Mutex = 1

What is the maximum no. of processes that may present in CS at any point of time?

Page No.
Date

& (V) → will up it 0 to 1

Down → will down the value of mutex 1 to 0 or 0 ka toh Blocklist

mutex = ✗∅ ✗∅✗0

(I) P_1 - 9 → (II) P_2 → (V) P_3 → (III) P_{10} → P_1 P_{10} P_2 P_3 → (IV) CS

As P_{10} ke entry me Bhi up & Exit me Bhi up toh Bar Bar under Bahar Krke mutex ki value ko 1 Krr & Ek point ayega when all 10 Processes will be Inside (CS)

∴ Answer = 10 Processes

Down → (S) Ki value (1) hai toh Down (0) Krdega if (0) toh Blocklist me

Up → (S) Ki value (0) se (1) Krdega

* if P_{10} Ka Code is as follows:-

```
Repeat
V(mutex)
CS
P(mutex)
Repeat
```

(I) P_1 → (II) P_{10} → (III) P_2 → P_1 P_{10} P_2 → CS

Abh at max P_1 to P_9 Bahar Jaakar mutex ko 1 Krega & Ek aur process under ayega But at any moment at max Kewal 3 Processes hi Rh skti Hai (CS) me

→ Solution of Producer Consumer problem using Binary Semaphore & Counting

Case I → without preemption Case-II → Preemption (Yhi Problem ati Hai & there is need of Sync.)

Jo Process (S) ki value ko phle down krdega vhi Apega Bss Critical Section me so that Interference na ho Paye

Counting Semaphore → full = 0 = No. of filled slots
Empty = N = " " Empty "

Binary Semaphore = 1

Produce_item(item p)	N = 8	Consumer
I1) down (Empty)	0 X, 1 b, 2 c, IN 3 d, 4, 5, 6, 7	down (full)
I2) down (s)		down (s)
Buffer[IN] = itemp		itemC = Buffer[out];
IN = (IN+1) mod n;	IN = 3, out = 0	out = (out+1) mod n;
I3) up (s)		up (s)
I4) up (full)		up (Empty)

Case-1 → Without Preemption i.e phle pura producer chal Gya & than consumer chala or vice versa

Producer: Empty = 8 X X 5 → Total = N = 8
full = 0 X X 3
S = 1 0 X X 0 1
IN = 0 X 4

Consumer: out = 0 X 1

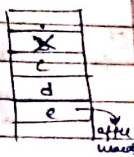
Synchronised

Case-2 → With Preemption (i.e preempt Hone Ke Baad Bhi Synchronise Hai)

Now,
Let's suppose producer ke I1 pe preempt ho gyi

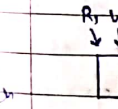
∴ Consumer vha se process shuru krega & as (s) ko voh down krega so producer chah kr Bhi Critical Sect^n me nhi Ghus Payega

Empty = X X 6 5 OUT = X 2
full = 0 X X 3
S = 1 0 X X 0 1
IN = X 5



Here as ek Baar ek process ghus gya (CS) me toh dusre ka Ghusna is impossible so Process here also gets synchronised in this Case

→ Solut^n of Reader - Writer problem using Semaphore



Same - data Base
R - W → Problem
W - R → "
W - W → "
R - R → No Problem
(Either to 1000 Readers)

Now, this problem is much more Related to a Data Base. Now as a Data Base is also Treated as the Critical Section so problem would arise with writer

2 Types of Users → Reader & Writer

Problem in same DB would arise in first 3 Cases as there could be manipulation happening in DB, But Reader to Jitne marzi aa Jaye Ek Time pe no problem hence, there is a need of Synchronization here also. Synchronising here done using Semaphore

(Read Count) int rc = 0

Semaphore mutex = 1 } → Binary
" db = 1 }

```
void Reader (void)
{ while (true)
  { down (mutex)
    rc = rc + 1
    if (rc == 1)
      then down(db)
    up (mutex)
    DB (CS)
    down (mutex)
    rc = rc - 1
    if (rc == 0)
      then up(db)
    up (mutex)
    Process data
  }
}
```

Entry

Exit

```
void writer (void)
{ while (true)
  { down (db)
    DB (CS)
    up (db)
  }
}
```

Now this code has Eradicated the problem this can be checked by checking for every case one by one

Case - I ⇒ R - W

R comes

rc = Ø X 0 db = X Ø 1

mutex = X Ø X Ø 1

DB R1 →

Now, as soon as (W) comes so as db = 0 so (W) Jaise hi ana chahega voh Block list me Chala Jayega & nhi aa payega ab Jaise hi Reader Bahar gya ~~toh mutex~~ toh mutex & db Fir se up ho gye & now writer can enter

Case - II ⇒ W-R & W-W

Similar Jab writer aya toh dB = 0 ho gya so Reader, or another writer will not be able to enter in Data Base Jaise hi Bahar Jayega & dB = 1 hoga toh hi koi Dusra Reader ya writer enter kr Payega

Case - III ⇒ R-R

R1 R2

↓ ↓

DB R1 R2

rc = Ø X 2 db = Ø X 0

mutex = X Ø X Ø 1

Reader ki Entry ki Kand" only with mutex so no problem in entring more than 1 R

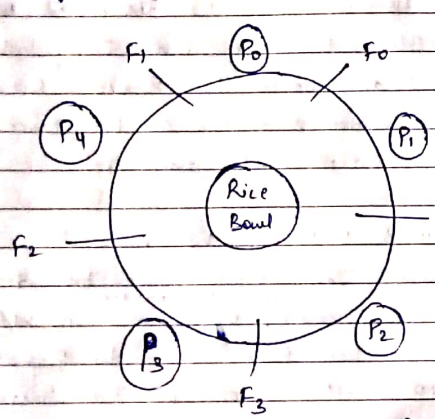
Reader vale exit code me bhut important Baat yeh hai ki voh ek Read Count pr Track RKhta hai & uspe condition Bhi lgata Hai. Hence Jab Tak 1 Bhi Reader Data Base me Betha hai tb tak db ki value 1 nhi writer Data Base me nhi

Page No.
Date

Ghus skta as uski Cond" db ke saath hai & agar I Krdiya toh writer ghuske problems create Kr skta hai

∴ All this process is Synchronised

→ Dining Philosopher's problem & Solution



5 Philosophers (P)
5 Forks (F)

2 Operat"s by Philosopher
- Think (Mutually
- Eat Exclusive)

Now, here if any philosoper (P) wants to eat than it must do following operat"s

i.e pick left Fork than pick Right Fork & then eat & then keep the forks Back

Page No.
Date

```
void philosopher (void)
{  while (true)
   {  Thinking ();
      take_fork (i)  // left fork
      take_fork (i+1 mod N) // Right Fork
      EAT ();
      Put fork (i)  // left      (N = No. of
        "  (i+1 mod N) // Right   Forks)
   }
}
```

Case-1 :- Har philosopher ek ek Krke Khaye

i.e P_0 then P_1

f_0 ↙ ↘ f_1 f_1 ↙ ↘ f_2

Case-2 :- P_0 -abh simultaneously

∴ f_0 ↙ ↘ f_1

P_1 aa gya & he needs f_1 & f_2 But f_1 toh P_0 le gya so here it becomes the problem

∴ if more than 1 philosopher coming to eat to ho skta hai ki Forks ka clash ho Jaye so leads to Race Condit"

Now, For this problem we need Array of Semaphores as a Solution

S[i] → five Semaphore ≡ five Phil.

S_0 S_1 S_2 S_3 S_4

↓ ↓ ↓ ↓ ↓

1 1 1 1 1

Page No.	
Date	

$P_0 \rightarrow S_0 \quad S_1$
$P_1 \rightarrow S_1 \quad S_2$
$P_2 \rightarrow S_2 \quad S_3$
$P_3 \rightarrow S_3 \quad S_4$
$P_4 \rightarrow S_4 \quad S_0$

$S_0 \; S_1 \; S_2 \; S_3 \; S_4$
$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$
$X_0 \; X_0 \; X_0 \; X_0 \; X_0$

isko down krna pdega for picking fork

* Now, is solution me bhi ek problem Hai i.e this solut^n can lead to deadlock i.e if a case arise that har philosopher apne left fork ko uthate hi preempt hogya toh case (c) ki value (0) ho Jayegi Hence, every philosopher will eventually apna second Fork uthane ke chakkar me Block list me chala Jayega Hence saari processes Blocked

Deadlock condition

$P_0 \rightarrow S_0 \quad S_1$
$P_1 \rightarrow S_1 \quad S_2$
$P_2 \rightarrow S_2 \quad S_3$
$P_3 \rightarrow S_3 \quad S_4$
$P_4 \rightarrow S_4 \quad S_0$

$S_0 \; S_1 \; S_2 \; S_3 \; S_4$
$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$
$X_0 \; X_0 \; X_0 \; X_0 \; X_0$

Block list
$P_0 \; P_1 \; P_2 \; P_3 \; P_4$

Now, iske Problem Ka yeh Solution hai Ki P_0 to P_{n-2} taki ki processes ko as it is chalne do But Bas P_{n-1} (last Process) ko order ulta krdo i.e Kewal us process ko phle kho ki Right Fork phle uthaye & then left fork uthaye (Zaruri Nhi ki Kewal last Process ko hi ulta krna Hai kisi bhi Ek ko kr skte Hai)

Page No.	
Date	

$P_0 \rightarrow S_0 \quad S_1$
$P_1 \rightarrow S_1 \quad S_2$
$P_2 \rightarrow S_2 \quad S_3$
$P_3 \rightarrow S_3 \quad S_4$
Inverted → $P_4 \rightarrow S_0 \quad S_4$

$S_0 \; S_1 \; S_2 \; S_3 \; S_4$
$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$
$X_0 \; X_0 \; X_0 \; X_0 \; 1$

Block list
P_4

↳ Abh yeh is 0 ko Down krne ki koshish me Block ho Jayega

& Now Kyunki $S_4 = 1$ Hai toh Baaki processes Block nhi hongi & Deadlock cond^n Nhi Ayegi

∴ (N-1) Processes
wait (Take_fork (i))
wait (" (i+1 mod N))

Nth Process
wait (takefork (i+1 mod N))
wait (takefork (i))

These changes in Code

```
void philosopher (void)
{   while (true)
    {   Thinking ();
        wait (take fork (i))
        Wait (take_fork (i+1 mod N))
        EAT ();  (CS)
        Signal ( Put fork (i))
        Signal ( Put fork (i+1 mod N))
    }
}
```

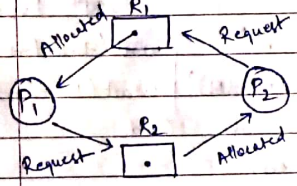
→ Deadlock

Mutual Exclusion → To Resources ho Rhe voh mutual Exclusive manner me ho ki Jab Tak Ek Process Resource ko use Kr Rha Hai toh koi aur Nhi Kr Skta us Resource Ko

* If 2 or more processes are waiting on happening of some event, which never happens, then we say these processes are involved in deadlock then that state is called deadlock

MHB Ki Agar Event Ke Hone Ka wait Kr Rhe hai for ∞ time & ho hi Nhi Rha Voh Kaam toh deadlock hai

eg-



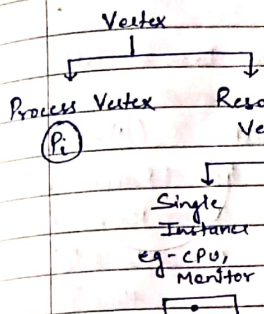
Now is Closed loop process me P_1 pr R_1 Hai But Request for R_2 to get Execute whereas P_2 pe R_2 Hai But Requests for R_1 to Execute so Never Ending process hence deadlock

4 Necessary & Sufficient Conditions for a system to be in deadlock

1) Mutual Exclusion
2) No Preemption (Process Ke Bich me Preemption Nhi Honi Chahiye)
3) Hold & wait (Holding the process & waiting for the Requested Resource to allocate) (And Jab Tak ek current Process ki Execution na ho Jaye agli Study nhi Hogi)
4) Circular Wait (loop Formation ho Rha ho Waiting for Requested Resource ka) (Just like above loop)

→ Resource Allocation Graph (RAG)

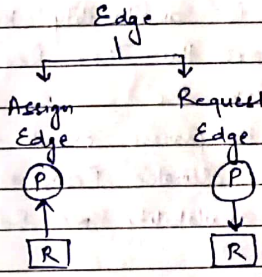
Graph to Represent state of the system



(all the processes & Requests in a system are Represented by Vertex of the RAG)

(only 1)

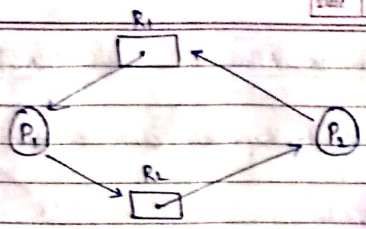
(They are in multiple numbers)



These would come Questions in which you would be provided with Systems Represented by RAG & we have to tell whether they are in the Condit" of deadlock or not

∴ 2 ways either tell by Observing all the 4 necessary & Sufficient Conditions or Follow the proper procedure

a)



Now, for checking Just Draw a Table (matlb pta kr)

	Allocate		Request	
	R_1	R_2	R_1	R_2
P_1	1	0	0	1
P_2	0	1	1	0

(1) → matlb us Resource ke liye voh Kaam ho Rkha Hai & (0) means nhi Hua)

∴ R_1 R_2

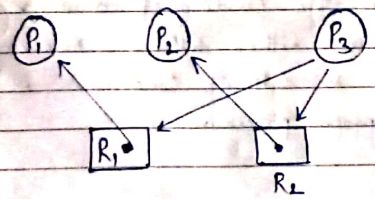
Availability (0, 0) → Now we have to Check ki is availability ke sath esi Koi Bhi process hai Jiski Request pori krke usse execute krna ho

also Jaise hi Koi process Execute Hoti Hai vaise hi uski allocated Resources Free ho Jate Hai & availability me Bd Jaati Hai

Now, as idhar Koi Bhi Process ki Request puri Nhi ho Rhi toh isska matlb all the processes has to wait for ∞ time so

State of Deadlock

b)



	Allocate		Request	
	R_1	R_2	R_1	R_2
✓ P_1	1	0	0	0
✓ P_2	0	1	0	0
✓ P_3	0	0	1	1

R_1 R_2

Availability (0, 0) → As it is availability se P_1, P_2 execute ho sakte so P_1 Execute ho Jayega

Hence Allocated Resources Free

Availability $(\overset{R_1}{0}, \overset{R_2}{0})$

1 0

(1, 0) → P_2 Execute

Allocated Resource Free

Availability $(\overset{R_1}{1}, \overset{R_2}{0})$

0 1

(1, 1) → Now P_3 can be Executed with this availability so it will be Executed

R_1 R_2

availability (1, 1)

0 0

(1, 1)

Process Execution ⇒ $P_1 \rightarrow P_2 \rightarrow P_3$

As all the processes have been Executed & no process has to wait till ∞

∴ No Deadlock

Waiting

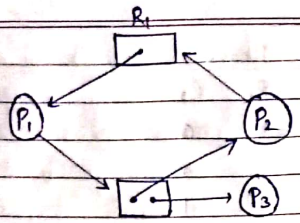
finite (even if it is 10000 years) Starvation

Infinite ∞ This is Deadlock

Single Instance RAG

- if RAG has Circular wait (cycle) then always deadlock
- if RAG has no cycle then no Deadlock

Q)



Multi Instance RAG

	Allocate		Request	
	R_1	R_2	R_1	R_2
(ii) ✓ P_1	1	0	0	1
(iii) ✓ P_2	0	1	1	0
(i) ✓ P_3	0	1	0	0

Availability ⇒ $(0, 0)$ (R_1 R_2) → P_3 hi Execute ho sakta

$0\ 1$

⇒ $(0, 1)$ → P_1 Execute ho jayega

$1\ 0$

⇒ $(1, 1)$ → P_2 Krdo

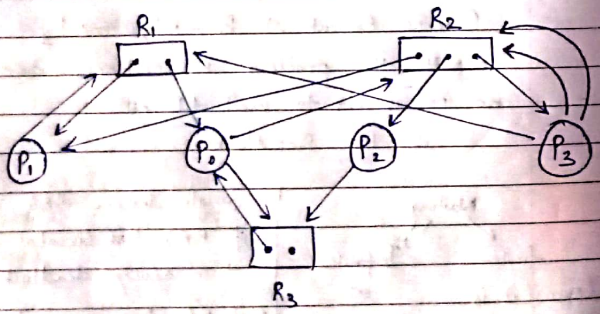
$0\ 1$

⇒ $(1, 2)$ → Final Availability

as every process executed Hence, No deadlock

$P_3 \rightarrow P_1 \rightarrow P_2$

Q)



	Allocate			Request		
	R_1	R_2	R_3	R_1	R_2	R_3
(ii) ✓ P_0	1	0	1	0	1	1
(iii) ✓ P_1	1	1	0	1	0	0
(i) ✓ P_2	0	1	0	0	0	1
(iv) ✓ P_3	0	1	0	1	2	0

Availability ⇒ $(0\ 0\ 1)$ (R_1 R_2 R_3) → P_2 process hoga

$0\ 1\ 0$

⇒ $(0\ 1\ 1)$ → P_0 " "

$1\ 0\ 1$

⇒ $(1\ 1\ 2)$ → P_1 " "

$1\ 1\ 0$

⇒ $(2\ 2\ 2)$ → P_3 " "

$0\ 1\ 0$

$(2\ 3\ 2)$ → Final Availability

as every process got Executed Hence,

No deadlock

$P_2 \rightarrow P_0 \rightarrow P_1 \rightarrow P_3$

→ Various Methods to handle deadlocks

(Just like Ostrich Storm se Bchne ke liye sand me ho mooh daal leti hai & ignore kr deti Hai)

i) Deadlock Ignorance (Ostrich method)

ii) Deadlock prevention

iii) Deadlock Avoidance (Banker's Algo) (Mainly to avoid by checking if Process is safe or unsafe)

iv) Deadlock detection & Recovery

i) In this case we simply ignore if a deadlock State occurs & Suprisingly this is the most practiced Technique in the market. Windows &

→ Banker's Algorithm (Jo Steps ya process Humne RAG me deadlock Identify Krne ke liye Kiya tha)

Process	Allocation A B C	Max Need A B C	Available A B C	(Request) Remaining Need (Max need − Allocation) A B C	
P_1	0 1 0	7 5 3	3 3 2	7 4 3	P_1
P_2	2 0 0	3 2 2	5 3 2	1 2 2	P_2
P_3	3 0 2	9 0 2	7 4 3	6 0 0	P_3
P_4	2 1 1	4 2 2	7 4 5	2 1 1	P_4
P_5	0 0 2	5 3 3	7 5 5	5 3 1	P_5
	7 2 5				

(As Processes goes on Executing the available Resources goes on Increasing)

Total A = 10, B = 5, C = 7

Sequence ⇒ $P_2 \rightarrow P_4 \rightarrow P_5 \rightarrow P_1 \rightarrow P_3$

(Sequences Bohot Tarah ki ho SKti Hai But Saari Ek Ek Krke Execute Hoti Chali Jayengi & Deadlock occur Nhi Hoga)

Questions toh is algo pe 2 Tarah ke possible Hai Ek to using RAG aapko system provide Krvaya tha & dusra Tabular form me apko Sari details Bta Rkhi Hai & apko abh Chize Check Krke Btani Hai

(Remaining need ≤ Curr. Availability) for all Resource types to get Executed

Q) A System is having 3 Processes each Require 2 units of Resources 'R'. The minimum no. of units of 'R' such that no deadlock will occur —

a) 3 b) 5 c) 6 (d) 4

Now, 3 Processes

P_1 P_2 P_3 (Har ek ko execute Hone ke liye 2 Resources Chahiye so for No deadlock we can simply say to give 2 Resources to Each Process)

Hence, $3 \times 2 = 6$ → This is an answer

But we need minimum Resources & when we check for deadlock then we check for every case that kisi Bhi Case Chahe Resources kisi Bhi manner me distribute ho Bhi ho deadlock nhi ang Chahiye

eg - i) if R = 2

P_1 P_2 P_3

1 1 1 1 1 1

(i) (ii) (iii) Execute Hone pr Resources Vapis Jate Rhenge

But if

P_1 P_2 P_3

1 1

(This occurs deadlock) (it could occur)

ii) if R = 3

P_1 P_2 P_3

1 1 1

(No deadlock)

But if

P_1 P_2 P_3

1 1 1

(deadlock)

iii) if R = 4

P_1 P_2 P_3

1 1 1 1

(No deadlock in any case)

Hence, Here the answer is (4)

* Hence, iske answer ke liye sari processes ko Jitne Resources ki Requirement thi execute ke liye un sab Processes ko use Ek Ek Resource km allot kro & then in total Ek (+1) Resource aur add krdo (which will not let the deadlock to occur)

eg - i) $\begin{pmatrix} P_1 & P_2 & P_3 \\ 1 & 1 & 1 \end{pmatrix} + 1 = 4$ Resources Minimum so that deadlock never occur

(2 ki Requirement thi as sbko)

ii) $P_1 \rightarrow 3$ $P_2 \rightarrow 4$ $P_3 \rightarrow 5$

Ans ⇒ $\begin{pmatrix} P_1 & P_2 & P_3 \\ 2 & 3 & 4 \end{pmatrix} + 1 = 10$ Resources Minimum so that deadlock never occurs

Q) Consider a System with 3 Processes that Share 4 instances of same Resource type. Each process can Request a max. of 'K' Instances. The largest value of 'K' that will always avoid deadlock is —

K → Here is no. of Resource to be provided to a Resource so that it could get Executed & Hume deadlock Bhi occur Hone Nhi deni so hai case Bhi Check Krna pdega deadlock Ke liye

For these type of Questions, let

P_1 P_2 ... P_n (Processes)

d_1 d_2 ... d_n (Resources Reqd. by each process to Get Executed)

Page No.
Date

Now, for cond" of max. Resources but still deadlock
⇒ $(d_1-1) + (d_2-1) + (d_3-1) \cdots (d_n-1)$
Sbko unki zaruroto se 1 kam diya
∴ Cond" Arises that (R) is total Resources

$R \leq \sum_{i=1}^{N} d_i - N$ —— deadlock

$R > \sum_{i=1}^{N} d_i - N$ —— deadlock Free

⇒ $\boxed{R+N > \sum_{i=1}^{N} d_i}$ This cond" must satisfy so that deadlock do not occur

Convinient Way

Total Resources + Total Processes	>	Total Demand

Now, For this Quest" as (K) so, Total demand = 3K

∴ $4 + 3 > 3K \Rightarrow 7 > 3K$

$K < 2.33 \Rightarrow \boxed{K=2}$ Answer

eg- 5 Processes 6 Resource (K=?)

$5 + 6 > 6K$

$11 > 6K \Rightarrow K < 1.88$

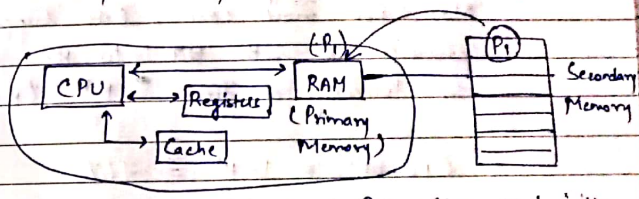
$\boxed{K=1}$ (Reqd. Ans)

Page No.
Date

→ Memory Management in OS (Managing Primary Memory)

Degree of Multiprogramming → Secondary memory se 1 nhi Balki Jyada se Jyada Program ko Primary memory (RAM) me laane ki Koshish kro so that CPU ki max. utilization ho & vo idle na Bethe (Jyada processes ane se interference na ho voh Bhi Chiz OS handle Krta Hai)

Is portion me mainly Hume memory management Primary Memory ki krni Hai (mainly RAM)



Secondary memory ki speed Bohot Kam Hai isliye direct CPU se attach nhi ki Jaati whereas RAM ki speed is Kaafi High so Hai Ek Chiz vhi Interface Kr Rha Hota Hai
Primary memory ki speed Bohot Jyada Hai But Capacity Kam Hoti Hai Jyada me Cost Bohot ↑

Process
CPU ↙ ↘ I/O

Isi vjah se Multiprogramming krte Hai so that Jis Time pe I/o ke liye kai process Jaye us time CPU Idle na Bethe & utilizat" Bd Jaye

* Dhang se samajhne ke liye taking example

eg- (1). RAM Process

4MB 4MB

No. of Processes = $\frac{4MB}{4MB} = 1$

Lets suppose, (K) → I/O operat^n (Percentage of Time)

CPU utilizat^n → (1−K) (Percentage of Time)

if K = 70%.

∴ CPU utilizat^n = 30%.

(ii) Process = 4MB CPU = 8MB

No. of Process = 2

Now, for worst case, lets suppose Dono processes ek sath I/O ke liye gye

So Percentage of Time = K^2 (K=70%)

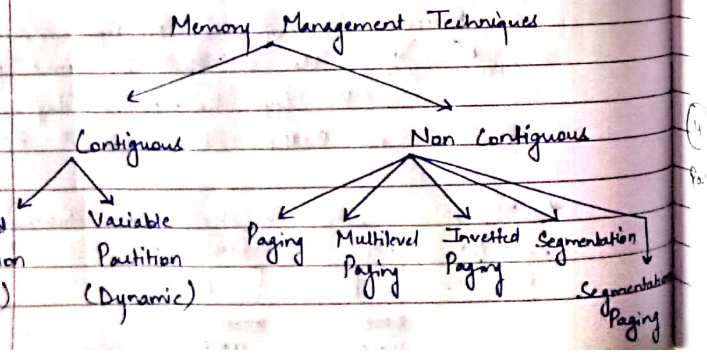
CPU util. = $1-K^2 \cong 76\%$.

(iii) Process = 4MB CPU = 16MB

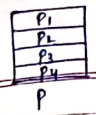
No. of Process = 4

for worst Case = K^4 (K=70%)

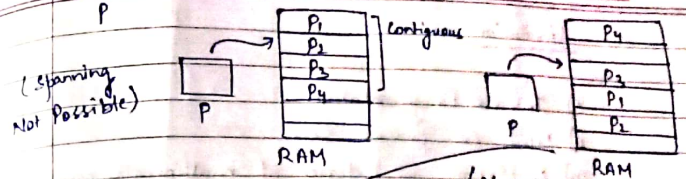
CPU utilizat^n $\cong 1-K^4 \cong 94\%$



eg- Lets suppose Ek Process ke 4 subparts hai

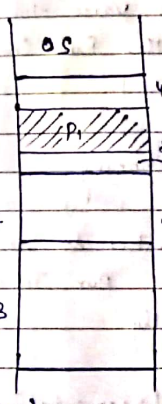


→ so in sab parts ko contiguous memory allocat^n Technique me memory me Contiguous allocat^n Chahiye



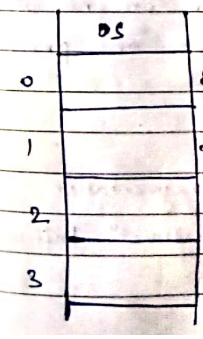
Is alag alag Jagah subparts ka allocat^n is called Spanning

→ Fixed Partitioning (Memory ki pre defined Partitioning Kri Hoti Hai for the allocat^n & statically)



(4) Partitions

Lets say (P_1) Process of 5MB aya toh voh 8MB vale Sect^n me mount Hogi But usme Bche hue 3MB waste ho Jayenge as us sect^n me koi aa nhi skta so voh wasted isse khte hai Internal Fragmentation



Static Partition

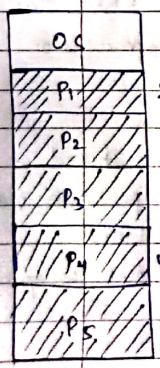
- No. of partitions fixed
- Size of each partition may or may not fixed
- Contiguous allocat^n so spanning not allowed

Limitations

i) Internal Fragmentation
ii) Limit on process size
iii) Limitation on degree of multiprogramming (i.e ek time fixed no. of processes hi aa skti hai)
iv) External Fragmentation → Mtlb if 4 Processes allocate hui & unki wastage is 2MB, 1MB, 1MB, 2MB Respectively. Now, if a process of 5MB wants to mount toh voh nhi possible hai mount ho paana as vaise toh Gaps ka Jode toh 6MB Space hai But Contiguous 5MB Space Nhi hai.

→ Variable or Dynamic Partitioning

Here partitioning ka number & size are not fixed yeh sab chize Run Time pr decide hoti hai dynamically

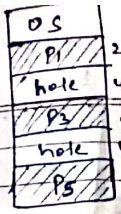


- No Internal Fragmentat"
- No limitation on no. of processes
- No limitation on process size

Limitations

i) External Fragmentation
ii) Allocat"/Deallocat" Difficult

Agar kai process deallocate ho Jaye toh voh ek hole bn Jaati hai utne hi space ka



So, Abh agar us hole ki space se Bda kai process aya eg- 5MB toh as contiguous allocat" honi chahiye so voh hole Bekaar ho Jayega. Hence, Compaction krni pd skti hai i.e Holes ko ikatha kr do ek Taraf

→ Memory Allocat" Algorithms

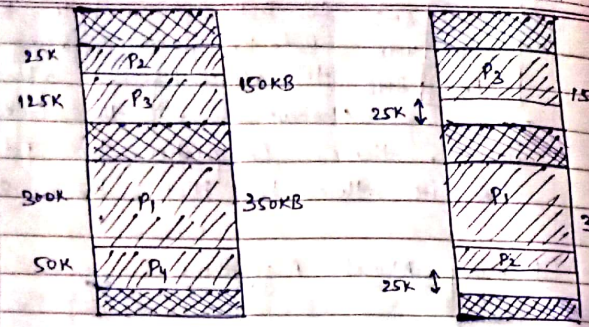
i) First fit → Allocate the first hole that is Big enough to accomodate
ii) Next fit → Same as First fit But start Search from last allocated Hole always (mtlb Jo Hole last allocate hua tha search pointer vha se shuru krega search krna) (as vha ek pointer point kr rha hoga)
iii) Best fit → Allocate the smallest hole that is Big enough (Gap must be minimum)
iv) Worst fit → Ek dum ulta; Allocate largest hole (Gap must be maximum)

* Best fit me kbhi kabar itne chhote holes Rh Jaate hai ki aur processes shayad hi aa ske leading to external fragmentation
* Most convinient is First fit

Requests from processes are 300K, 25K, 125K, 50K Respectively (isi order me Requests aayegi) the above Request could be satisfy with

a) Best fit but not First fit
c) Both
b) First fit but not Best fit
d) None

First fit



(Now P4 accomodate Nhi ho Payega Yha)

Best Fit

Q) "BEST FIT"

Request No.	J1	J2	J3	J4	J5	J6	J7	J8
Request Size	2K	14K	3K	8K	6K	10K	7K	20K
Usage time	4	10	2	8	4	1	8	6

Calculate the time at which J7 will be Completed —

a) 17 c) 20

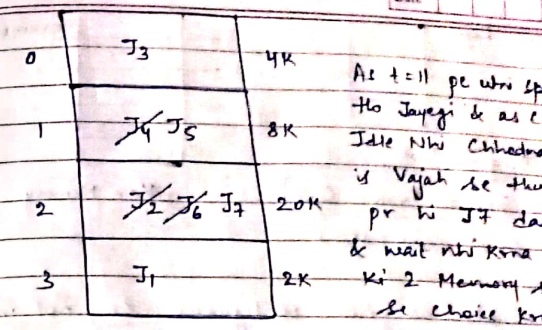
b) 19 d) 37

Idhar Jo memory given hai that is to be taken as fixed Partitioned se krke diye thi Partitions.

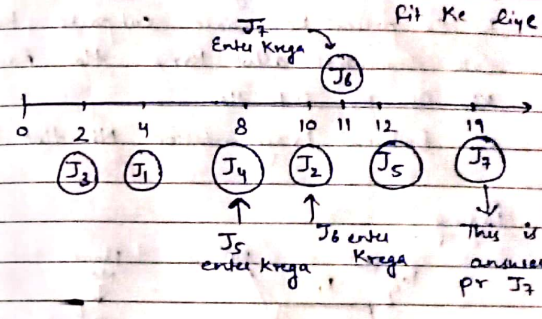
And also thai Ek Process/Job ka usage Time diya hua hai so, Ek Timeline maintain krni pdegi

(Also as fixed Partitioned so chahe Holes ho ya Nhi Bchi kuchhi Space me kuchh aur Nhi aa skta)

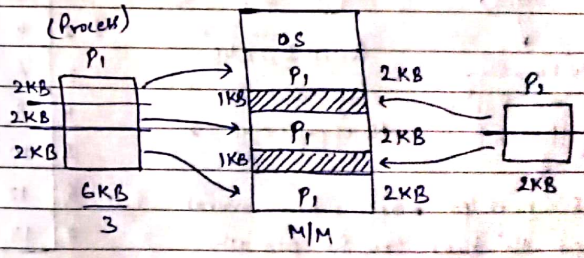
M/M



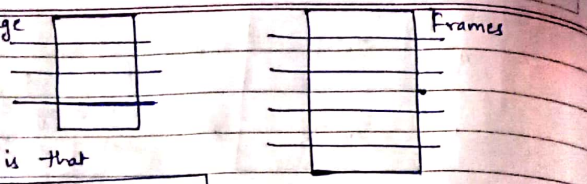
At t=11 pe utni space khali ho jayegi & as CPU ko Idle Nhi Chhedna so is Vajah se thrne-11sec pr hi J7 daalna hai & wait nhi krna t=12 kr ki 2 Memory spaces me se chaise krr Best fit Ke Liye.



Non Contiguous Memory Allocation



Now, here process ko divide krke suppose me Break krke memory me daal Rkha thi part wise. Therefore, now division bohot hi costly process hai agar Run time pr kiya Jaye to Bohot hi Time lgega (as Time consuming) so hum log phle hi Dono chize ko division krke Rkhte Hai



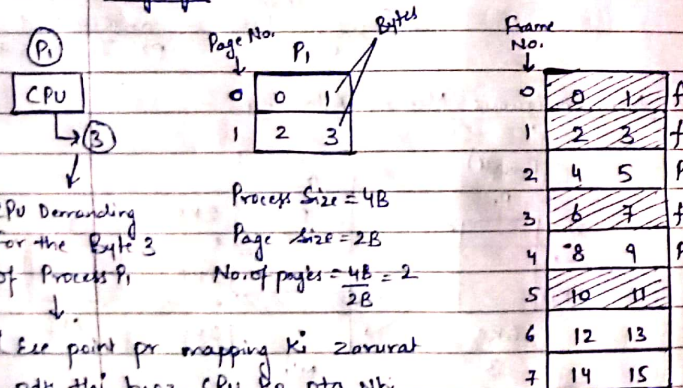
Imp. Thing is that

Page Size = Frame Size

This Cond^n must Hold

also as non-contiguous so not important Ki Ek process ke saare subparts ko ek sath Sequential allot krna hai Randomly Khi Bhi Kr Skte hai
∴ External Fragmentation Nhi Hoti isme

→ Paging



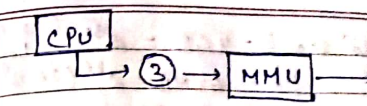
CPU Demanding for the Byte 3 of Process P1

(Ese point pr mapping Ki zarurat pdti hai bcoz CPU Ko pta Nhi Ki paging hui hai but usse to voh vala data Chahiye so mapping se pta Chalega ki main memory me voh data exactly Kha gya hai)
(as isthar Byte 3 main memory ke Byte 9 par Hai)

Process Size = 4B
Page Size = 2B
No. of pages = $\frac{4B}{2B}$ = 2

M/M Size = 16B
Frame Size = 2B
No. of Frames = 8

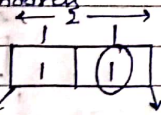
MMU → Memory management unit ⊻ Hai process Ka Ek Alag MMU Hoga which is Responsible for mapping



Page Table of P1

0	2 → Page 0 is Frame me Hai
1	4 → Page 1 " " " "

* Logical Address

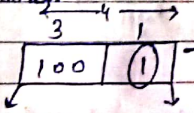


as here Page No. is $d = d^1$ ∴ 1 Bit is Reqd. Similarly Page offset = d^1 ∴ 1 Bit is Reqd. Hence logical address is of 2 Bits

Here, this logical address is just like virtual Address of Microprocessor

(# Page offset Value = Logical offset Value (Always hold))

* Physical Address

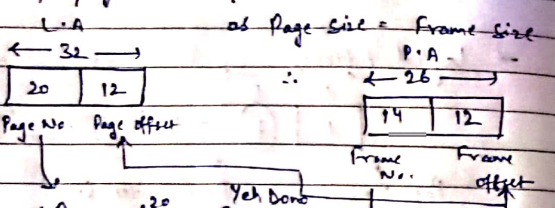


As Frame No. i.e total no. of Frames are 8 = d^3 ∴ 3 Bits Reqd. to Represent de Frame Size = d^1 ∴ 1 Bit " " "

Now, if we see toh logical add^r = 3 ka Physical Add^r = 9 Hence, proper mapping/conversion Hogyi

Memory is Byte Addressable
(Smallest unit Byte se memory ko access krte hai)

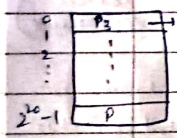
Q) LAS (Logical add^{n} space) = 4 GB = 2^{32} (32 Bits ka Logical Address)
PAS (Physical " ") = 64 MB = 2^{26} (26 Bits ka P.A)
Page size = 4KB = 2^{12} (12 Bits ka page size/offset)
No. of pages = ? No. of Frames = ?
No. of Entries in page Table = ?
Size of page Table = ?



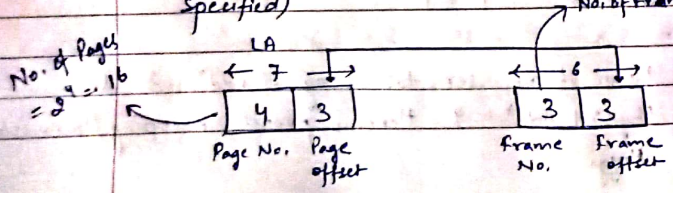
as Page size = Frame size
∴ No. of Pages = 2^{20}
Yeh Done Some Hota Hai
No. of Frames = 2^{14}

* No. of Entries in Page Table = No. of Pages
= 2^{20}

* Size of Page Table = 2^{20} × 14 Bits
↓ Utne Entries of Pages
↓ Har Frame ko Represent Krne ke liye 14 Bits Reqd.

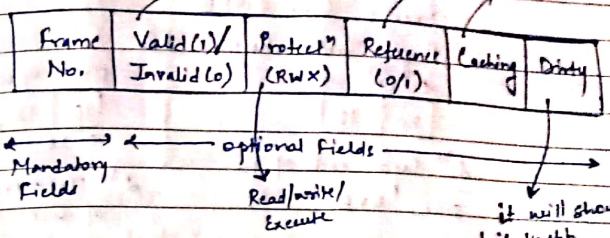


Q) Consider a system which has LA = 7 Bits
PA = 6 bits, Page Size = 8 words/Byte then Calculate no. of pages & no. of frames
(By default word = Byte; Jab Tak nothing is Specified)



as 8 size ka page size $8 = 2^3$ ⇒ 3 Bits to Represent

→ Page Table Entry



(Now, Hota kya hai CPU sbse pehle Cache ko hi Check krta hai for the data so agar koi data Baar Baar call kr rha hai toh hum us data ko Cache memory me save krlete hai so that Shortest Path se CPU whaye & time ghat Jaye isse Khte hai Caching)

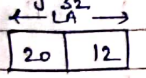
Reference means ki kya yeh page PMe Bhi aya hua hai ki Nhi (1) means aya hua hai & method used here is LRU (Least Recently used)

Caching can't be used with the Bank Transact^{n} system as it needs to be updated & nhi to voh some info ko hi Bdhata Jayega

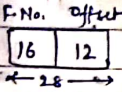
→ 2 Level Paging

Sometimes Page Table ka size itna Bda ho Jata hai ki voh Frame size ko Bhi Exceed kr Jata hai Hence, Voh fit Nhi hoga So hum Ek Page Table ka Bhi Index type Bnate hai leads to 2 level Paging

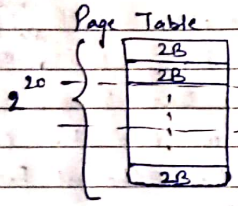
eg. PAS = 256 MB
LAS = 4GB
Frame size = 4KB
Page Table Entry = 2B



2^{20} Pages each of size 4KB



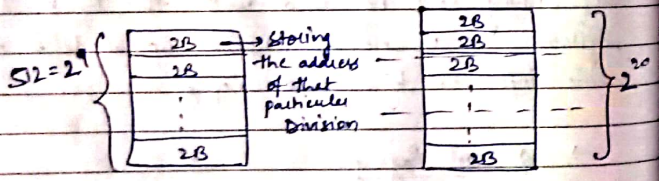
2^{16} Total frames each of size 4KB



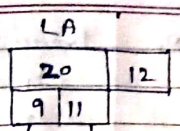
PT Size = $2^{20} \times 2B$
= 2MB > 4KB

Now the process ka alag P.T Hai Jo frame me hi Store Hota Hai But yha is PT ka size is greater than 4KB (size of frame)
So, we further divide page Table (PT)

$\therefore \frac{2MB}{4KB} = 512 = 2^9$ Divisions

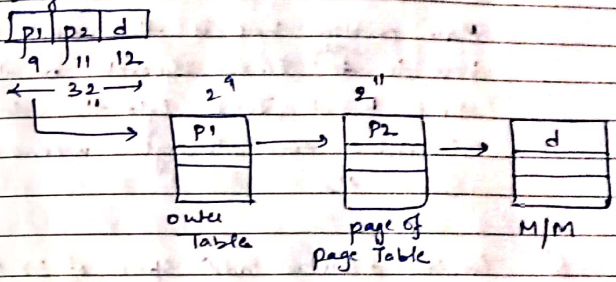


Now $2^9 \times 2$ = 1KB = Size < 4KB
This outer Table is also described by the LA Itself



2^9 → Divisions
$\frac{4KB}{2B} = 2^{11}$ (That means no. of Pages in That whole division)

Address Translat" Scheme



→ Inverted Paging

1) Each Process has its own PT
2) PT will be in M/M

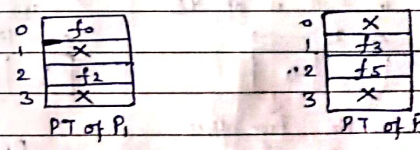


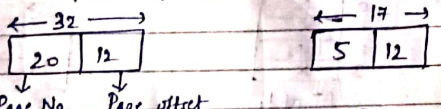
Abb Yeh Tarike se memory Bohot Jyada Lgti Hai Hence, memory Bchane ke liye we make Global PT Jisme we store every thing about frame no. & Page no. of each process ek dum ulte format me. Upar vale process me Frame no. is inside, Global PT me frame no. is outside

Global PT

Fr. No.	Page No.	Process ID
0	p_0	P_1
1	p_1	P_2
2	p_2	P_1
3	p_1	P_3
4	p_3	P_2
5	p_2	P_3

Main problem here is that each time we have to apply linear search on it & hence time Bohot Jyada lgta hai so not successful even though memory Bohot Bchti hai

Q) Consider a virtual add^n Space of 32 Bits & Page Size of 4KB System is having a RAM of 128 KB. Then what will be the Ratio of page Table & Inverted Page Table Size if each Entry in Both is of 4B



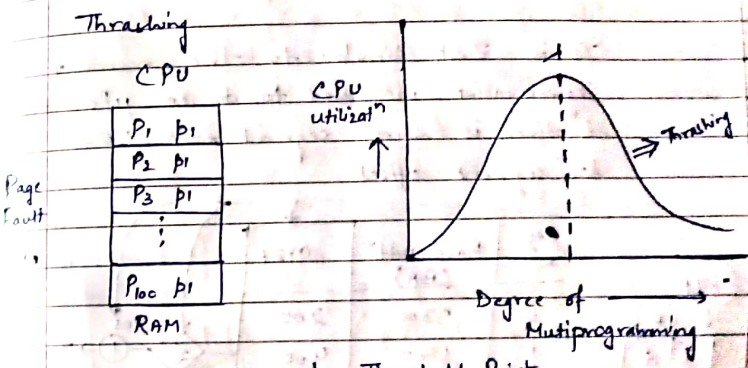
Ans. Size of PT $\propto$ No. of Pages
& " " Inverted PT $\propto$ No. of Frames

$\therefore$ Ratio = $\frac{\text{Size of PT}}{\text{Size of Inverted PT}} = \frac{2^{20} \times 4B}{2^5 \times 4B} = 2^{15}:1$

→ Thrashing

We Know, with Increase in Degree of multiprograming CPU utilization Increase but at page fault also occurs if let's say Hai ek process ka page 1 RAM me present Hai & CPU wants Page 2 for Execut^n. Hence, usse thik Krne Ke liye Page Fault service time lgta Hai.

Therefore Ek Point aata Hai Ki agar usse Jyada multiprogramming Krdi toh CPU Ki utilizat^n Decrease Hone Shuru ho Jati Hai Called Thrashing



A – Threshold Point

Remedies

i) M/M Size Increase

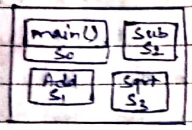
ii) Long Term Scheduler i.e Scheduler ko kho Ki apni speed thodi kam Krte

Page Fault Simply means Ki jis page Ki demand ayi voh page main memory me Hai hi Nhi Hence, isse fix Krne me kaafi time lgta Hai as Hard disk se uthake laana pdta Hai usse & Hard " is very slow so that time is called Page Fault Service time

→ Segmentation

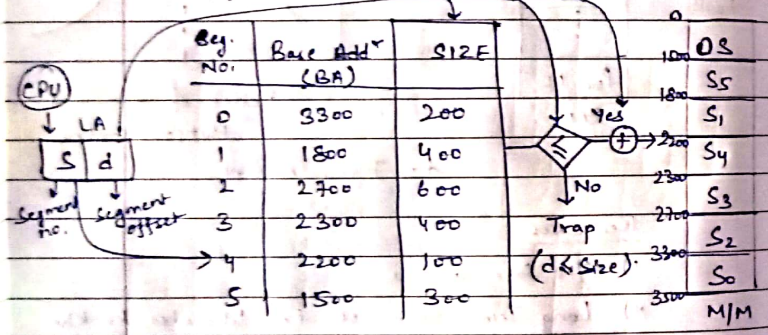
Paging is simply kisi bhi process ko divide kr dena but Segmentation user point of view se process ko divide krta hai eg- Addition ka ek whole subject, subtract" ka ek like that. So that Page fault doesn't occur

dividing into different Segments



Here Segments could be of variable size not supposed to be fixed

It is just similar to the memory Segmentation we used to do in µPs also deration is same segment no. (Base Address) & segment offset



Segment Table

LA — MMU — PA

Is Tarah se hum Virtual add" vala kaam kr skte hai using B.A & segment offset & similar to µP idher bhi ek segment se dusre par nhi jaa skte

→ Overlay

It is method to even accomodate those processes whose size is larger than the size of main memory. This is done by partitioning the process but here no driver is alloted to segregate the partitions (the Required ones) in the main memory, here user has to do it. So don't use it in computers but used in Embedded systems

eg- M/M ka Biggest frame is 16 MB But process ka size usse bda to Reqd. Partition ko lekar aye M/M me so that pura process hi accomodate ho jaye

(i) Consider a 2 Pass assembler pass 1 - 80 KB pass 2 - 90 KB ; Symbol Table = 30 KB ; Common Routine = 20 KB

At a time only 1 Pass is in use. What is min partition size Reqd. if overlay driver is 10 KB size

Pass 1	Pass 2	
80 KB	90 KB	80 KB
30 "	30 "	90 "
20 "	20 "	30 "
10 "	10 "	20 "
		10 "
140 KB	150 KB	230

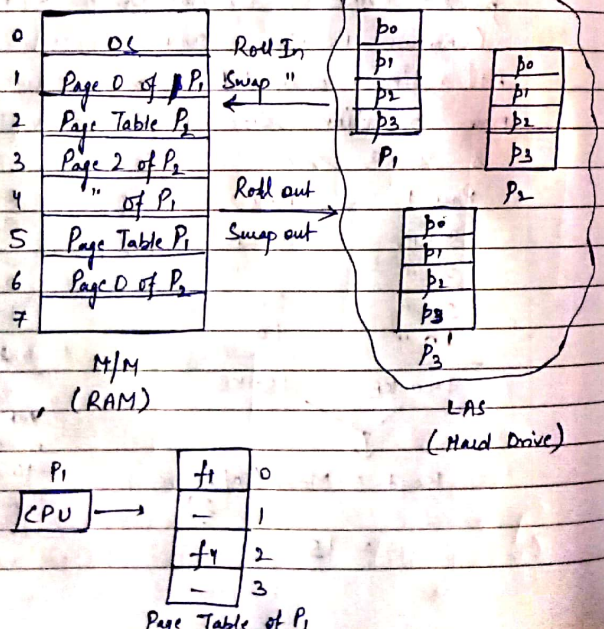
Reqd. = Max. (of Above 2)

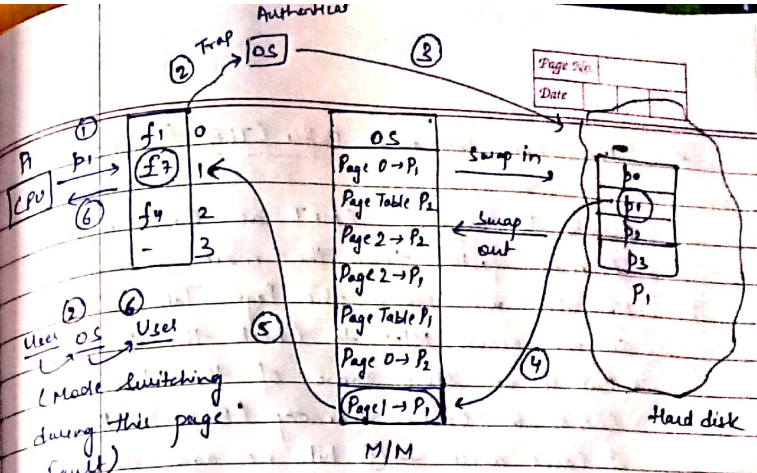
= 150

Yeh pura memory Reqd. ho gyi But thoda Min. Overlap so this is not valid

→ Virtual Memory

As memory size is limited but processes's size is increasing day by day & also no. of processes are Increasing.
∴ It is a concept through which it creates an illusion that Even the processes bigger than the main memory & much more no. of processes are being Accomodated in the limited main memory.
(Principle is that only Reqd. Pages of the processes ko main memory me lao & Jab unki need khtm toh unhe wapis le Jao & dusre le ao this swapping keeps on going)





Now, as when CPU demanded for p_1 of P_1 it wasn't present leading to Page fault. After that Page fault service occurred in the Given Chronology of steps numbered from 1 to 6.

Also, when page fault occur the mode switches from user to OS & then when Page fault service completes it again Returns to user mode.

EMAT (Effective memory Access time)

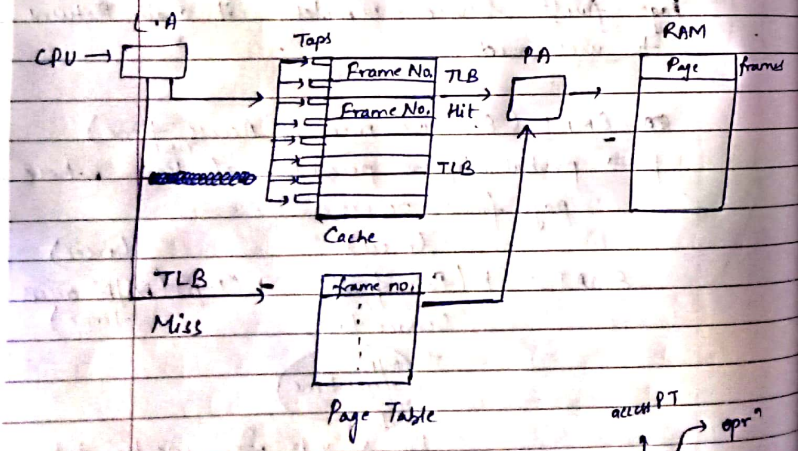
p → probability or percentage of $\frac{\text{Page fault occurs}}{\text{Total time}}$

$$EMAT = p\,(\text{Page fault Service time})_{(msec)} + (1-p)\,(\text{M/M access time})_{(msec)} + \text{M/M access time}$$

But as yeh nsec me hai toh as compared to above could be neglected isliye yeh term Nhi likhte

→ Translation Lookaside Buffer (TLB) (Cache)

Mainly to improve EMAT as Cache memory is more faster than the main memory.
Normally we store the Page Table in M/M
So we access M/M to check the Frame No.
* But if we store some of the Page Table Entries in Cache memory (Buffer) than
2 Cases arise → (i) hit i.e agar Jo maanga vhi mil Gya toh Sab kuchh fatafat access ho Jayega (ii) Miss i.e agar Jo maanga uski Entry Cache me nhi Hai toh PT access krke frame dekhna Pdega
Let x → time to access PT from M/M
x → To fetch on that M/M
∴ if no caching is there than,
EMTA = x + x = 2x ——— (1)



EMAT = (TLB + x) + (TLB + x + x)
Hit → opr^n
Miss → access PT → opr^n

as cache memory is faster so EMAT gets improved when no. of Hits Increase. But as Cache is limited Hence, saari PT ki Entries usme Nhi daal skte. But thodi Bohot daalte Hai

Q) A Paging scheme using TLB. TLB access time is 10 ns & main memory access time takes 50 ns. What is EMAT (in ns) if TLB hit Ratio = 90% & there is no page fault

∴ EMAT = Hit (TLB + M/M) + Miss (TLB + M/M + PT)

$= \frac{90}{100} \times (10 + 50) + \frac{10}{100} (10 + 50 + 50)$

= 65 ns

→ Page Replacement Algorithm

1) FIFO
2) Optimal Page Replacement
3) Least Recently Used (LRU)
4) Most Recently Used

Here, we would be given with some frames & a series of Requests (had to be considered in given order only)
Now, if there is a hit we just copy previous Entries & if a miss / Page Fault than we Bring that Entry from Hard Drive & Replace a entry on the Basis of algorithm.

* - Page fault

Page No.
Date

Q) FIFO

f_3			1	1	1	X	0	0	Ø	3	3	3	Ø	2	2
f_2		0	0	0	Ø	3	3	Ø	2	2	2	Ø	1	1	1
f_1	7	7	Ø	2	2	2	Ø	4	4	Ø	0	0	0	0	0
	*	*	*	*	Hit	*	*	*	*	*	*	Hit	*	*	Hit

Reference String ⇒ 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0

Now FIFO me ya kisi me bhi Jab Tak Frame khali thi Tab Tak to us khaali vale me Bharte Jao But Jaise hi Replacement ki Baari aye to ~~usi~~ us order me kro i.e ki Jo pehle aya use pehle Replace kro eg for eg - f_1 pehle Replace fir f_2 fir f_3 fir f_1 & so on

No. of Hits = 3 No. of Page faults = 12 (Miss)

Hit Ratio = $\frac{3}{15} \times 100$

Miss Ratio = $\frac{12}{15} \times 100$

* There is a problem / Anomaly with FIFO that if we Increase no. of Frames than page fault also Increase (whereas Ghatna Chahiye tha) called Belady's Anomaly

Q) Optimal Page Replacement (Replace the page which is not used in longest Dimension of time in Future) (MHb Sab given options me se Future me Jo Sbse door Hai use Replace krdo)

f_4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
f_3				1	1	1	1	X	4	4	4	4	4	X	1	1	1	1	1	1
f_2			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_1	7	7	7	7	X	3	3	3	3	3	3	3	3	3	3	3	Ø	7	7	7
	*	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit	Hit

Reference String ⇒ 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

as idhar out of 4 options agar hum future me check kre to 7 Sbse door hai isiliye 7 Replace hua

Similar manner me Replacements hui

No. of Hits = 12 No. of miss = 8

Q) Least Recently Used (LRU) (Chaaro me se left Hand side me Jo Sbse door ho usse Replace krdo) (as usse Recently use nhi kiya)

f_4			2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
f_3		1	1	1	1	X	4	4	4	4	4	X	1	1	1	1	1	1	1
f_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_1	7	7	7	X	3	3	3	3	3	3	3	3	3	3	3	Ø	7	7	7
	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit	Hit

Reference String ⇒ 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

idhar out of 4 left me 7 was Sbse door So voh Replace Hogya

No. of Hits = 12

" " Miss = 8

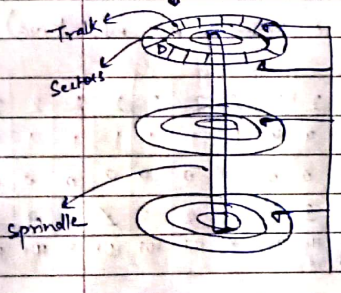
Replace the least Recently used Page in Past

Q) Most Recently used (MRU) (Replace the most Recently used page in past)
(Chalo me se jo sbse (near) paas me ho past me usse Replace krdo)
(Ek Dum ulta Criteria of LRU)

For same string, MRU se
No. of hits = 8 No. of miss = 12

Ref string => 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
as sbse paas me 0 hai past me toh usse Replace krdiya

→ Disk Architecture



(P) (S) (T) (S)
Platters → Surface → Track → Sectors → Data (D)

(Normally Each Platter have 2 Surface up & down)

(M)
Memory It can Carry like as Follows

M = 8 × 2 × 256 × 512 × 512 KB
(Platter) (Surface) (Tracks) (Sectors) (Each Sector could contain this much data)

M = 2^{40} B = 1 TB ; No. of Bits = 40 Reqd.

M = P × S × T × S × D

Platters unitrolled ghumte hai & Actuator arm unitrol age piche chalte hai ek time pe

→ Disc Access Time

1) Seek Time - Time Taken by R/W Head to Reach desired Track. (Current Track se Desired)
(Hum Avg. value lete hai i.e Best Case ki hum usi Track pe hai & of worst Case ki Bohot Jyada Travel Krna pde)
2) Rotation Time - Time Taken for one full Rotation (360°)
3) Rotation Latency - Time Taken to Reach desired sector
(half of Rotation Time)
4) Transfer Time - Data to be Transfer / Transfer Rate

Transfer Rate (Data Rate) = No. of Heads (Surface) × Capacity of one Track × No. of Rotat°s in one second

(Kitne sector ka data Transfer krna hai & ek sec me Kitna data Transfer Hojata hai)

Disc Access Time = ST + RT + TT + CT + QT
(ST + RT + TT: Yeh Zaruri hai yeh to hai hi)
(CT + QT: These only when Given the NW toh not necessary)

→ Disk Scheduling Algorithms (DSA)

Goal - To minimize the seek Time

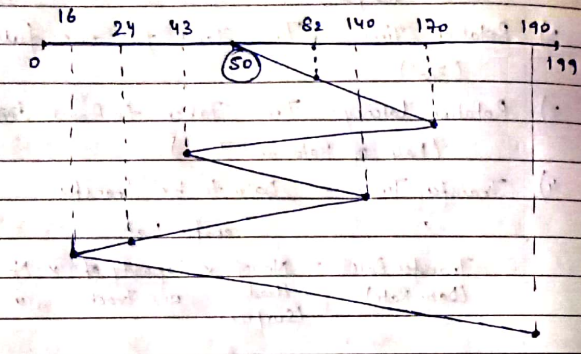
DSA
- FCFS (First come First Serve)
- SSTF (Shortest Seek Time First)
- SCAN
  - Look
  - CSCAN (Circular SCAN)
  - CLOOK (Circular Look)

No starvation but as yeh direction change ki parvah nhi karta toh seek time jyada hota hai

Page
Date

Q) FCFS

A disk contains 200 Tracks (0-199) Request Queue contains Track no. 82, 170, 43, 140, 24, 16, 190 Respectively. Current Position of R/W Head = 50. Calculate total no. of Tracks movement by R/W Head.

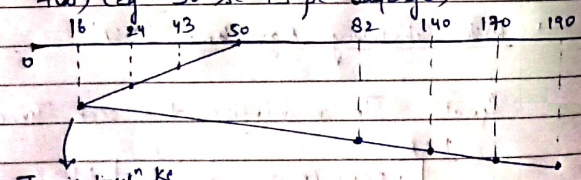


Jo pehle aye use pehle Process karo no matter of direction

Total no. of Track movements = (170 - 50) + (170 - 43) + (140 - 43) + (140 - 16) + (190 - 16)

∴ = 642

Q) SSTF (Same Question) (Matlb ki kisi particular time pe us point pr jaana hai jo us pt. se nearest hai) (eg - 50 se 43 pe jayenge)



Jab is direction ke saare Points khtm ho jaye toh dusre pr Jo

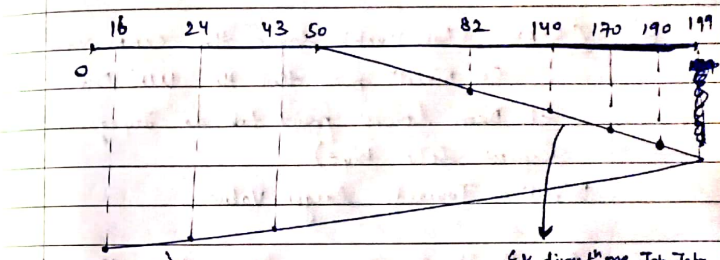
Page
Date

Total movements = (50 - 16) + (190 - 16) = 208

Response time accha hai but starvation hoti hai & as search krne me time lgta hai isiliye Time Consuming hai

* If R/W head takes 1ns to move from one Track to another then total time Taken = 208 × 1ns = Total Mov. × Time for 1 Mov.

Q) SCAN (Same Question)



Direction → Towards larger values (Yeh Given hoti hai)

Ek direction me Jab Jata hai toh us Direction Ke Final Track value Tak Jaata hai Bhale hi us value ki Request hai ya Nhi toh 199 tak Jayega

But final move me us direction Ke last point Tak nhi Jayega Balki sirf us direction Ke last Requested Point Tak Jayega

Total Mov. = (199 - 50) + (199 - 16) = 332

Total Time Taken = 332 × 1ns