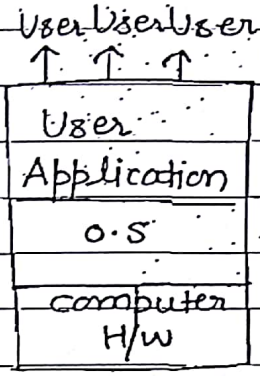
Operating System is an interface b/w user & computer hardware.




```
main
{
  printf("Hello");
}
```

→ it internally calls write() system call inorder to communicate with the monitor.

System Call - It is the request made by the user progrm to the operating system in order to get any kind of service.

- An operating system is also called as resource allocator.

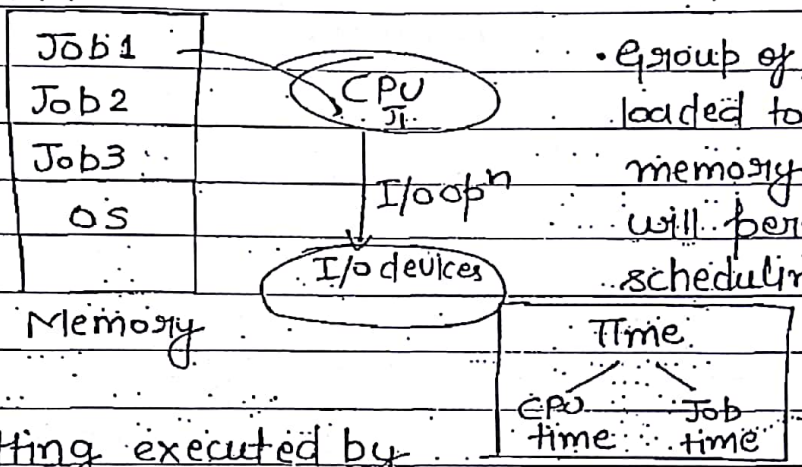
Resources
- Hardware Type — Eg: Devices, Memory
- Software Type — Ex- files,

Goals of Operating System-

1. The primary goal of OS is convinience (Ease to use)
2. The secondary goal of OS is efficiency

Types of OS-

1. Batch operating System



- Group of job is loaded to main memory & OS will perform scheduling of job onto the CPU.
- Let Job1 is getting executed by CPU, CPU instruction are getting executed. if Job1 has to perform I/O operation, it will leave the CPU & go to I/O devices & execute I/O operation. till that time, CPU is idle & still, it donot execute any other job. Because Batch system follows the strategy that if the job is computed completely (ie including CPU time & I/O time) than only another job is ~~scheduling~~ scheduled on CPU.

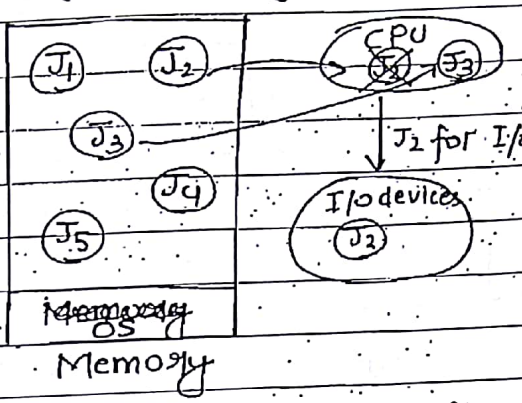
- It increases CPU idleness Hence decreases throughput of the system.

Throughput the No. of job completed for unit time is called as throughput of the system.

- Ex:- IBM OS/2

Date: / /
Page No.

2. Multiprogramming operating system-



- OS will sched job on the CP

- Let J_2 get scheduled to CPU by OS. After executing some mstn, it will leave for executing I/O operation. When CPU will became idle, the another job is assigned to CPU for its execution so that idleness is decreased.

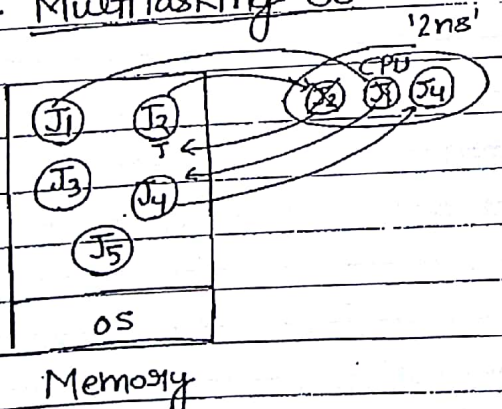
1. if the job is leaving CPU to perform I/O opn than another job which is ready for execution will be scheduled on the CPU.

∴ we have

1. Increased CPU utilization
2. Increased CPU throughput of the system.

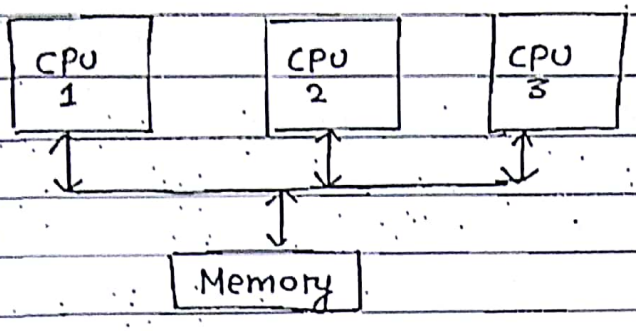
Ex- Windows, unix, Linux.

3. MultiTasking OS.



1. Multitasking is an extension of multiprogramming OS.
2. Jobs will be executed in the time sharing mode.
3. any job is executing for a particular time stamp (Let 2ns) for 2ns J_1 is executed & then it is send back to memory & J_2 is send to

Ex- Window, Unix, LINUX.



- More than one CPU in a single system.
- Increased throughput as job getting executing in unit time will be more.
- Reliability/Fault tolerant system

- Economical- Buying 3 system with 1 CPU is costlier than purchasing a single system with 3 CPU ∴ it is cheaper.

Ex- UNIX

5. Multi Core System-

6. Real Time System- the systems which are strict deadly time bound are called as Real time os.

(there should be no delay while exⁿ inⁿ of program)

Two types- 1. Hard Real time - Missile system, Satellite system
2. Soft Real time - Banking sector

In hard real time, minor delay are not accepted but in soft real time, minor delay are accepted.

Ex- Sx works, Vx Works, RTOS
ISRO DRDO

```
{
  int pid;
  pid = fork();
  if (pid < 0)
  {
      printf("child process creation failed");
  }
  else if (pid == 0)
      printf ("child process");
  else
      printf ("Parent Process");
}
```

1. Fork is a system call used to create the child process.
2. Fork returns negative value if the child process creation is unsuccessful.
3. Fork returns value 0 ; ~~if the child~~ to the newly created child process.
4. Fork return positive value (process id of child process) to the parent process.
5. When the child process is created by using Fork() system call the new m/m location will be allocated to the child process.
6. the parent & child processes will have same virtual address but physical addresses will be different.

Date : / /
Page No.

for 1 fork

```
main()
{
  fork();
  printf("Hello");
}
```

it will create a child process

```
main()
{
  pf("Hello");
}
```

Parent process

C_1 child process (fork is viewed from it)

for 2 fork

```
main()
{
  fork();
  fork();
  printf("Hello");
}
```

```
main()
{
  fork();
  printf("Hello");
}
```

C_1

```
main()
{
  pf("Hello");
}
```

C_2

Parent process

```
main()
{
  pf("Hello");
}
```

```
main()
{
  fork();
  fork();
  fork();
  pf("Hello");
}
```

```
main()
{
  fork();
  fork();
  pf(Hello);
}
```

```
main()
{
  fork();
  pf(Hello);
}
```

```
main()
{
  pf(Hello);
}
```

```
main()
{
  fork();
  pf(Hello);
}
```

```
main()
{
  pf(Hello);
}
```

```
main
{
  pf(Hello);
}
```

```
main()
{
  pf(Hello);
}
```

13 15 (d)

5

Date : / /
Page No.

Process Management- Program- passive: static

Process Concept-

Definition- the program under execution is called as process

for a program to be in execution-

1. it should resided in the main M/M.
2. it must be occupying the CPU to execute the inst^n.
3. Active and dynamic.

Process is having various attributes-

(i) Process Id
(ii) Process state
(iii) Program Counter
(iv) Priority
(v) General purpose registers Ex- Accumulator etc.
(vi) List of open files → the files used by process currently
(vii) List of open devices → the devices used by process.
(viii) protection information - whether any protection by os or not.

(i) Process Id- It is the unique identification No which is assigned by the OS at the time of process creation.

(ii) Process state- the process state contains current state information of a process where it is residing.

(iii) Program Counter- It contains the address of the next inst^n to be executed.

(iv) Priority- the parameter assigned by the OS at the time of process creation.

Ex- 9, 2, 7 etc.

- the context of the process will be stored in PCB (Process Control Block).

Process id	Process state
Program ctr	Priority
GPR	LOF
LO Devices	Protection Info

→ PCB7 (-It implies that it is a process 7 PCB)

- Every process will have its own PCB
- PCBs of the processes will also get stored in main M|M.
- PCB of any processes get implemented with the help of Double linked list.

- Process is having various states:
(i) New state
(ii) Ready State
(iii) Running State
(iv) Wait or Black state
(v) Termination or completion state
(vi) Suspend Ready state
(vii) Suspend Wait or Suspend Block state

- the various operation will be performed on the process.
(i) creating a process
(ii) Scheduling a process
(iii) Dispatching a process
(iv) Killing or Terminating
(v) Executing
(vi) Suspend
(vii) Resume

Date : / /
Page No.

Process State Diagram:-

1. Initially Process will be in New state (i.e process is under creation or being created
2. Once the process is created, it will enter into Ready state
3. So many process will be in ready state for getting execut from them, a process is selected & that will be dispatched on to the running state.
4. When a process is in run state, it is occupying CPU & executing the instⁿ of process & performing the CPU time activities.
5. At a time, only a single process can be in run state.
6. While run If the running process require I/O operation, that will move on to wait or Block state.
7. In the wait state also, there will be multiple No. of process. it means, multiple processes will perform I/O operation simultaneously
8. if the running process completes, it goes to termination state.

Multiprogramming is categorized into two types-

(i) Non-preemptive
(ii) Preemptive / Multitasking / Time sharing

Non preemptive- process is not forced to block in mid of its execution. it ends when its complete its exⁿ.

Preemptive- When program process is forcefully block in mid of execution because of priority or expiration of time quantum.

. All the multitasking is indirectly multiprogramming, Not all multiprogramming is multitasking

Date : / /
Page No.

- A process is preemped because of priority also. if a low priority process is running & a high priority process come in ready state, it is made to stop.

- When the process is in the ready, wait, running, or block state it is residing in the main memory.

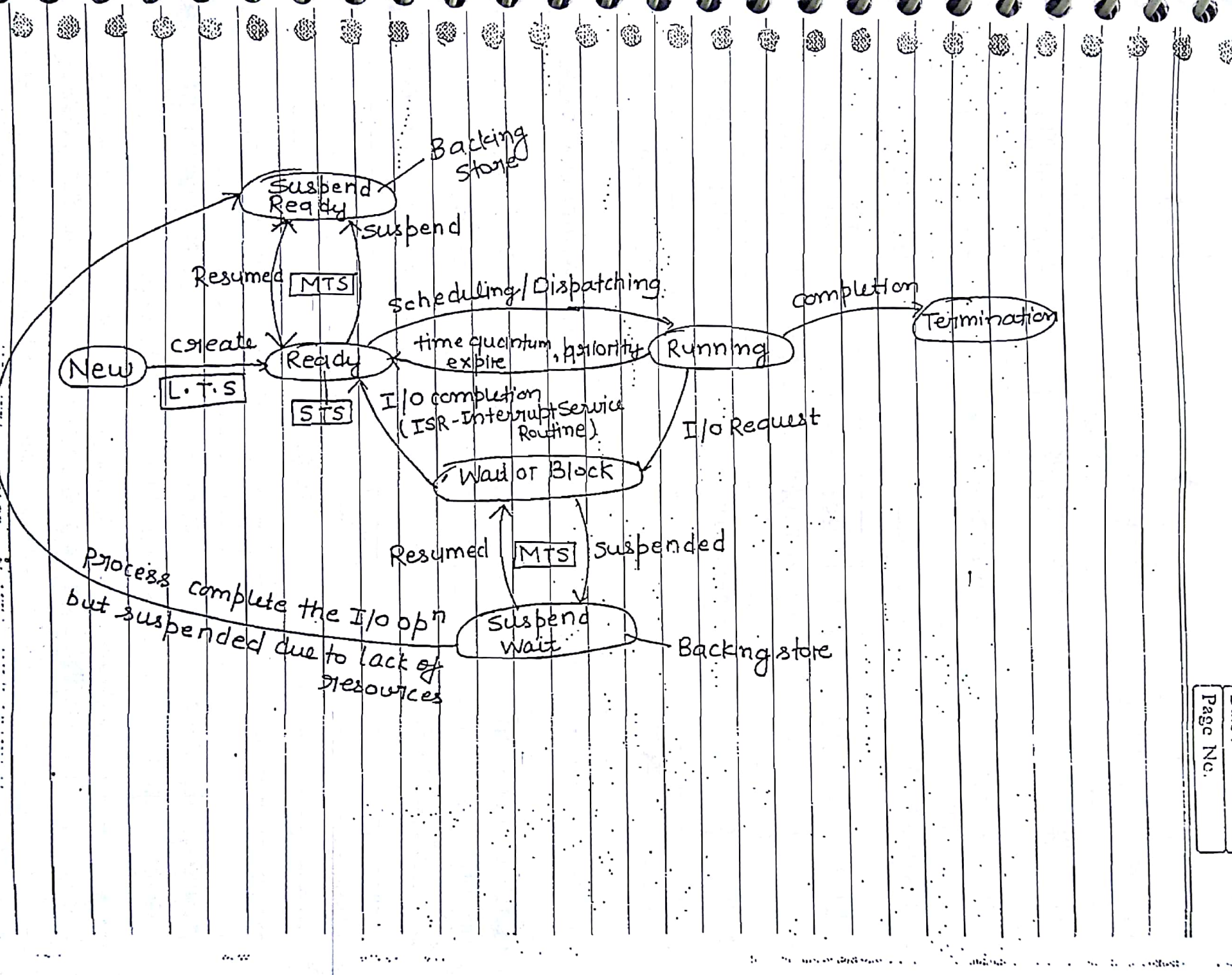
- When process is preemped in mid due to expire of time quantum or priority, it goes to ready state.

- When process is in block state or wait state, & I/O op^n complete it move to ready state.

- Since so many process are in ready state if the resources are not sufficient to manage the process in ready state than some of the processes will be suspended & they will be moved on to suspend ready state.
- When process is in suspend ready state, it is residing in the "Backing Store (Secondary Memory)"

- If so many process come for I/O operation & resources are not sufficient to manage the process in wait state, so process are suspended & moved to suspend wait state, residing in the backing store.

- Process is in wait state, Let P_L is the process with low priority & present in wait state & executing the I/O op^n. Let the process completes the I/O operation & about to go to ready stak but at same time, a high priority came & due to defficiency of resource P_L is moved to suspend wait state. Since it has completed the op^n, it is sucklenly moved to suspend ready state from suspend wait state.


Backing store
Suspend Ready
suspend
Resumed
MTS
scheduling / Dispatching
completion
Termination
create
New
L.T.S
Ready
STS
time quantum expire, priority
Running
I/O completion (ISR - Interrupt Service Routine)
I/O Request
Wait or Block
Resumed
MTS
suspended
Process complete the I/O opn but suspended due to lack of resources
Suspend Wait
Backing store

Page No.

in the main memory at any point of time is called as degree of multiprogramming.

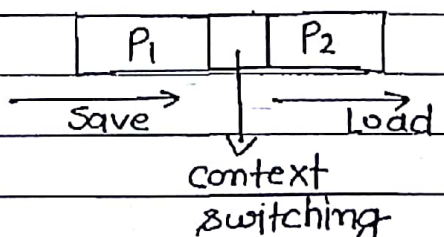
Processes are categories into two types
1. CPU bound processes
2. I/O bound processes

CPU Bound Process- The processes which require more amount of CPU time are called as CPU Bound process.
- these processes will spend more time in the running state.

I/O Bound Process- the processes which require more amount of I/O time are called as I/O bound processes
- these processes will spend more time in the wait or Block state.

- Each & every time when the process is moving from one state to other state, the context of the process will change.

Context Switching-



- Saving the context of one process & loading the context of another process is called as context switching.

- If the context of the process is more than the context switching time will also increase which is undesirable.
- the context switching time is considered as overhead for the system.

Date: / /
Page No.

Scheduler- In OS, there are 3 different schedulers-

(i) Long term scheduler/Job Scheduler - the LTS is responsible of creating & bringing the new processes into the system.

(ii) Short Term Scheduler/CPU scheduler - it is responsible of selecting one of the process from the ready state for scheduling on to the CPU or running state.

(iii) Mid Term Scheduler/Medium Term Scheduler. It is responsible of suspending & resuming the processes.
- the job done by mid term scheduler is called as swapping. (swapping from main M/M to Sec. M/M or vice versa)

- Scheduler are basically processes of OS.

Dispatchers - 1. it is responsible of performing context switching
2. it is also responsible of loading of selected job on to the CPU.

NOTE- In some special case, if there is only one process still that is called as context switching
Ex- Round Robin Scheduling with 1 process.

- Long Term scheduler controls degree of multiprogramming as more the process create by LTS more will be degree.
- the long term scheduler should select good comb of CPU & I/O bound process in order to get good throughput for system

Ques- Consider a system which has n-CPU processors than what is the minimum & max. No of process thay may present in Ready Running & Block state?

Ans-

	min	max
Ready	0	Any
Running	0	n
Block State	0	Any

- Any depends on maximum degree of multiprogramming in system

- Process is having different times-

(i) Arrival Time (A.T) - The time when the process is arrived into ready state is called as arrival time of the process.

(ii) Burst Time (B.T) - the time required by the process for its execution is called as burst time.

(iii) Completion Time - the time when the process is completing its total execution.

(iv) Turn Around Time - the time difference b/w completion time & arrival time.

$$\{TAT = CT - AT\}$$

(v) Waiting time - the time difference b/w turn around time & burst time.

$$\{W.T. = TAT - BT\}$$

Date : / /
Page No.

(vi) Response time (RT) - the time difference b/w first response & arrival time.

CPU Scheduling

Where- Ready State
Who Short Term Scheduler
When a) 1. Run → Termination - process goes from run to termination ∴ CPU become idle so a new process has to be assigned.
2. Run → Ready
3. Run → Wait
b) Wait → Ready Let a Low priority P_L is running & a new process with high priority complete I/O & move to ready state ∴ So P_L has to be preempted & P_H will be scheduled.
Let a high Priority P_H is running & P_L come to ready state from wait state. Scheduling Algo is applied & decision is to continue with P_H
c) New → Ready (to check if a high priority process had arrived or Not)

Goals- 1. Maximize the utilization of CPU & throughput of system
2. minimize the avg. TAT, avg W.T & avg R.T.

Algorithms- 1. First come First Serve (FCFS):
Criteria : Arrival Time
Mode : Non Pre-emptive

2.	1	3
3.	2	4
4.	3	6
5.	4	5

Find Avg TAT & Avg. WT = ?

{TAT = CT - AT}
{WT = TAT - BT}

CT1 = 2
CT2 = -

GANTT CHART-

P1	P2	P3	P4	P5
0	2 (CT1)	5 (CT2)	9 (CT3)	15 (CT4)	20 (CT5)

TAT1 = 2 - 0 = 2, TAT2 = 5 - 1 = 4 TAT5 = 20 - 4 = 16

TAT3 = 9 - 2 = 7 TAT4 = 15 - 3 = 12

Avg TAT = $\frac{2+7+4+12+16}{5} = \frac{41}{5} = 8.2$ unit

WT1 = 0 WT2 = 2 WT3 = 3 WT4 = 6 WT5 = 11

Avg WT = $\frac{21}{5} = 4.2$ unit

Ques-

P.No	AT	BT	CT	TAT	WT
1	6	5	31	25	20
2	3	4	12	9	5
3	4	8	20	16	8
4	1	2	3	2	0
5	5	6	26	21	15
6	2	5	8	6	1

Avg TAT & WT = ?

P4	P6	P2	P3	P5	P1
01	3	8	12	20	26	31

$\frac{79}{6}$ $\frac{49}{6}$

13.16 8.16

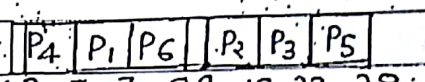
TAT = 13.16
WT = 8.16

Date : / /
Page No.

NOTE - if the arrival times of the processes are matching then schedule the process which has lowest Process Id.

Ques 4-

PNo	AT	BT	CT	TAT	WT
1	3	2	7	4	2
2	9	6	15	6	0
3	12	8	23	11	3
4	2	3	5	3	0
5	15	5	28	13	8
6	3	1	8	5	4
				42	17
				6	6
				= 7	2.83

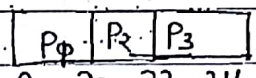


∴ TAT = 7

WT = 2.83

Ques 5-

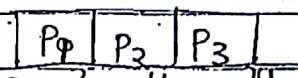
P No	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	2	22	21	19
3	2	2	24	22	20



TAT = 21

39/3 = 13 ∴ WT = 13.

P.No	AT	BT	CT	TAT	WT
1	0	2	2	2	0
2	1	2	4	3	1
3	2	20	24	22	2
					1



Avg WT = 1

TAT = 9

process (large BT) than it will have major effect on avg. waiting time of processes. this effect is called as the convoy effect.

2. Shortest Job first Algorithm-

Criteria: Burst time

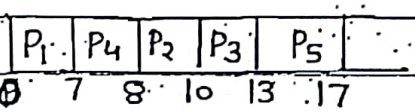
Mode: (i) Non preemptive

(ii) Preemptive

(i) Non preemptive-

Ques

PNo	AT	BT	CT	TAT	WT
1	0	7	7	7	0
2	1	2	10	9	7
3	2	3	13	11	8
4	3	1	8	5	4
5	4	4	17	13	9
				45	28



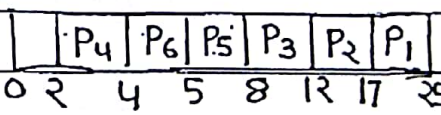
Avg WT = 5.6

Avg TAT = 9

Ques NOTE- If the burst time of processes are matching than schedule the process which has lowest arrival time. If AT also matches, choose for Id.

Ques

PNo	AT	BT	CT	TAT	WT
1	6	8	25	~~19~~	13
2	3	5	17	14	9
3	8	4	12	~~4~~	0
4	2	2	4	2	0
5	5	3	8	~~3~~	0
6	4	1	5	1	0
				~~43~~ 43	28



∴ Avg TAT = ~~8.5~~ 7.16

WT = ~~4.66~~ 3.33

~~8.5~~ ~~4.66~~

7.16 3.33

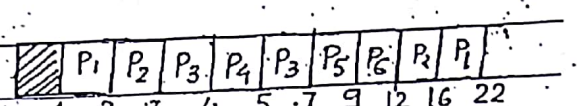
Date : / /
Page

(ii) Pre-emptive (Shortest Remaining Time First)

Criteria : Burst time

Mode : Preemptive

P No	AT	BT	CT	TAT	WT
1	1	16	22	21	15
2	2	8 4	16	14	9
3	3	3 2	7	4	1
4	4	1	5	1	0
5	5	2	9	4	2
6	6	3	12	6	3
				50	28
				8.33	4.83

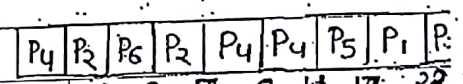


Avg TAT = 8.33

WT = 5

Ques-

P No	AT	BT	CT	TAT	WT
1	5	6	23	18	12
2	1	8 1 6	45	4	1
3	4	8	31	27	19
4	0	3 6 5	11	11	4
5	3	6	17	14	8
6	2	1 0	3	1	0
				75	44

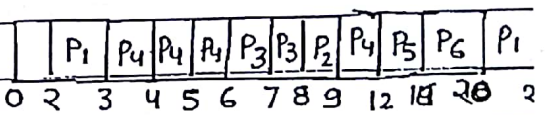


Avg TAT = 12.5

Avg WT = 7.33

Ques

PNo	AT	BT	CT	TAT	WT
1	2	8 1	27	25	17
2	7	10	9	2	1
3	6	2 1 0	8	2	0
4	3	6 5 4 3	12	9	3
5	5	4	16	11	7
6	5	4	20	15	11
				64	39



Avg TAT = 10.66

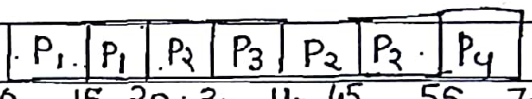
WT = 6.50

4	8	3	13	5

$\frac{33}{4}$ Avg TAT = 8.25 Ans

Ques

PNo	AT	BT	CT	TT	Total W.T of P2 using SRTF ?
1	0	20	20	20	0
2	15	25	55	40	15
3	30	10	40	10	0
4	45	15	70	25	10



WT = 40 - 25 = 15 Avg TAT = 23.75

WT = 6.25 Ans.

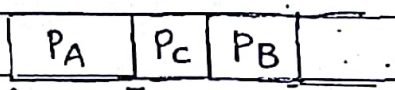
Ques: Consider 3 process A, B, C to be scheduled as SRTF Algorithm the process A is known to be scheduled first & when A has been running for 7 units of time, the process C arrived the process C has run for 1 unit of time than the process B arrives & completed running in 2 unit of time. what could be the minimum burst time of processes A, B & C?

$P_B = 2$

A > C

7 + x > C

1 + y > 2 Let y = 3 $\therefore P_C = 4$ for min.



$P_A = 12$

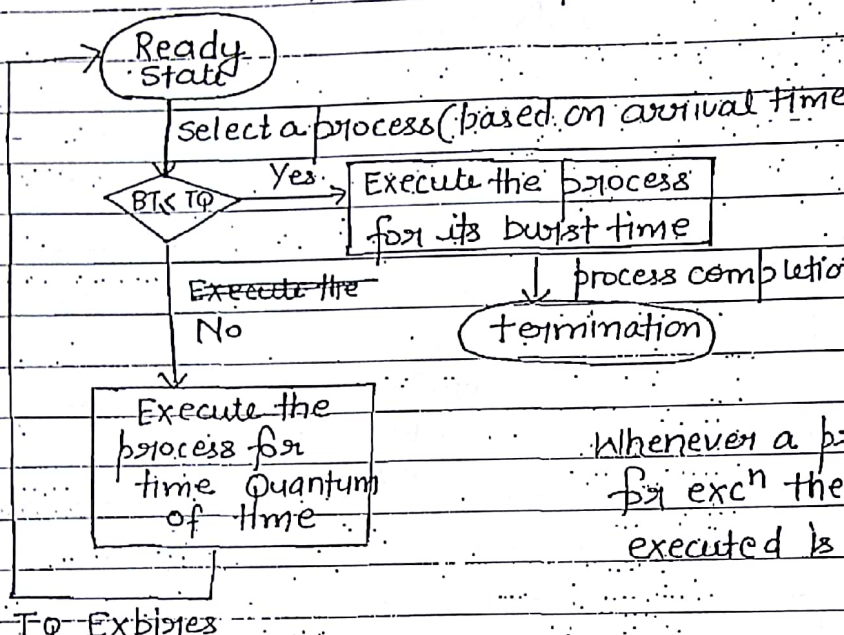
$\therefore$ A = 12 C = 4

Date : / /
Page No.

Round Robin Scheduling Algorithm

Criteria: Time Quantum or time slice / Arrival time

Mode: Preemptive

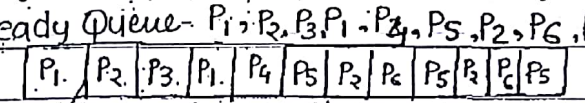


Whenever a process is scheduled for exc^n the max time it get executed is time quantum.

Ques- Let TQ = 2 Avg TAT & WT = ?

WT	PNo	AT	BT	CT	TT
4	1	0	4 2 0	8	8
12	2	1	5 3	18	17
2	3	2	2 0	6	4
5	4	3	1 0	9	6
11	5	4	6 4 2	21	17
10	6	5	3 1	19	13
44					65

Ready Queue- $P_1, P_2, P_3, P_1, P_4, P_5, P_2, P_6, P_5, P_2, P_6, P_5$



context switching

Avg TAT = ~~9 16~~ 10.83

WT = 7.33

Reponse time- ~~$P_1 = 0$ $P_2 = 2$ $P_3 = 4$ $P_4 = 8$~~ $P_5 = 9$

∴ ~~Avg RT = $23/6 = 3.83$~~

~~$RT_{P_1} = 0 - 0 = 0$~~ ~~$RT_{P_2} = 2 - 1 = 1$~~ $RT_{P_3} = 4 - 2 = 2$ ~~$RT_{P_4} = 8 - 5 = 3$~~ $RT_{P_5} = 9 - 6 = 3$ $RT_{P_6} = 13 - 11 = 2$

∴ Avg Response time = $10/3 = 3.33$.

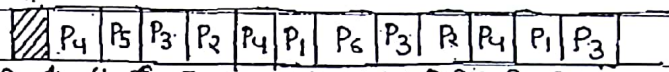
Date : / /
Page No.

TQ = 3

Ques-

PNo	AT	BT
1	5	52
2	4	63
3	3	74
4	1	93
5	2	20
6	6	30

RQ: P4, P5, P3, P2, P4, P1, P6, P3, P2, P4, P1, P3



	CT	TAT	WT	RT
1	33	27	22	10
2	27	23	17	5
3	33	30	23	13
4	30	29	20	0
5	6	4	2	2
6	21	15	12	12
	128	96	32	

Avg TAT = 21.33

WT = 16

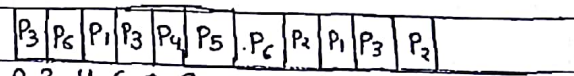
RT = 5.33

Ques-

P.No	AT	BT	CT	TT	WT	RT
1	2	10	16	14	11	2
2	5	31	19	14	11	8
3	0	80	18	18	12	0
4	3	40	9	6	5	5
5	4	20	11	7	5	5
6	1	40	13	12	8	1
			71	52	21	

TQ = 2

RQ: P3, P6, P1, P3, P4, P5, P6, P2, P1, P3, P2



Avg TAT = 11.83

WT = 8.66

RT = 3.5

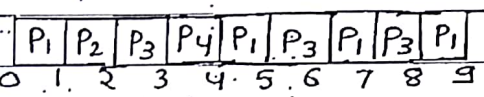
11.83 8.66 3.5

- In RR Algorithm, if the time quantum is less than no of context switches will increases & response time will be less.
- if the time quantum is large, than the no. of context switches will decrease & response time will be more.

Date : / /
Page No.

- if the time quantum is greater than all the burst time, than algorithm simply behaves like FCFS Algorithm.

Ques- consider a system which has four processes P_1, P_2, P_3, P_4 All arriving in ready queue in same order at the time 0. the Burst time requirement of these processes are 4, 1, 8, 1 respectively then, what is the completion time of process P_1 assuming RR scheduling with a TQ as 1?



(a) 7 (b) 8
(c) 9 (d) 10

Ques- Consider n processes sharing the CPU in RR fashion. the context switching time is 'S' than what must be time quantum 'q' such that, the no. of context switches are reduced but at the same time each process is guaranteed to get its job/turn after every 't' seconds of time?

$(n-1)q + ns = t$

$q = \frac{t - ns}{n-1}$

$(n-1)(q+s) = t$

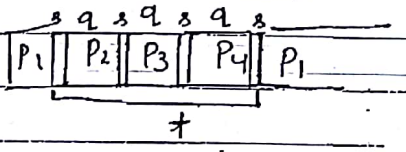
$n(q+s) = t$

$(n-1)(q) + ns = t$

$q = \frac{t - ns}{n-1}$

$q = \frac{t - s}{n}$

$t - \frac{sn}{n-1}$



Date : / /
Page No.

4. Longest Job First Algorithm- Job with longest burst time will be scheduled first

Criteria: Burst time

Mode : (i) preemptive

(ii) Non preemptive

• throughput of system decreases by LJF)

(i) Non preemptive-

Ques

P.No.	AT	BT	CT	TAT	WT
1	0	2	2	2	0
2	1	4	17	16	12
3	2	5	7	5	0
4	3	3	20	17	14
5	4	6	13	9	3
				49	29

Avg TAT & WT?

P1	P3	P5	P2	P4

0 2 7 13 17 20

Avg TAT = 9.8

WT = 5.8

Note- 1. if the burst time of processes are matching, than schedule the process which has lowest arrival time.

Ques-

PNo	AT	BT	CT	TAT	WT
1	4	4	22	18	14
2	3	3	29	26	23
3	5	8	18	13	5
4	1	2	3	2	0
5	6	4	26	20	16
6	2	7	10	8	1
				87	59
				14.5	9.83

	P4	P6	P3	P1	P5	P2

0 1 3 10 18 22 26 29

Avg TAT = 14.5

WT = 9.83

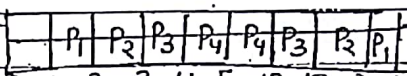
Date : / /
Page No.

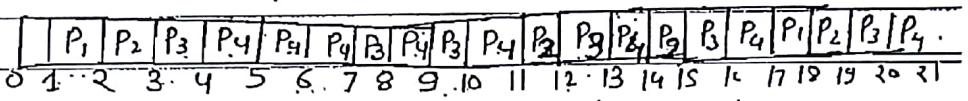
(ii) Longest Remaining Time first (Non-preemptive LJF)-

Ques-

PNo	AT	BT	CT	TAT	WT
1	1	21	21	20	18
2	2	4 3 2	20	18	14
3	3	8 5 4 3	17	14	8
4	4	8 7 6	12	8	0
		5 4 3 2	68	40	
			15	10	

Avg TAT =





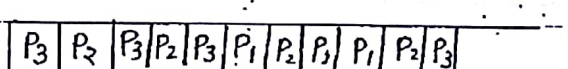
	CT	TAT	WT
1	18	17	15
2	19	17	13
3	20	17	11
4	21	17	9
		68	48
		17	12

Avg TAT = 17

WT = 12

Ques

P.No	AT	BT	CT	TAT	WT
1	0	2	12	12	10
2	0	4 3 2	13	13	9
3	0	8 4 3 2	14	14	6
				39	25
				13	8.33



Avg TAT = 13

Life Cycle of a process-

CPU time

I/O time

CPU time

I/O time

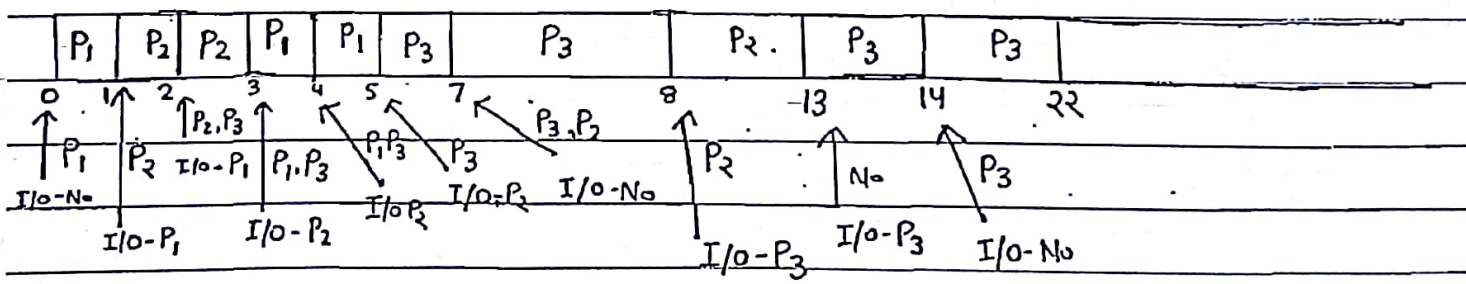
2. I/O - single process accessing I/O allowed
3. CPU, I/O → valid (1 access CPU & other I/O)
4. I/O, I/O, I/O → ~~Ad~~ valid
5. CPU, I/O, I/O → valid
6. CPU, CPU, I/O → invalid (as two processes cannot access CPU simultaneously)

Ques-

PNo.	AT	BT			
		CPU time	I/O time	CPU time	CT
1	0	10	20	10	5
2	1	20	40	5	13
3	2	3	6	8	22

what is the completion time of process P_1, P_2, P_3 using SRTF?

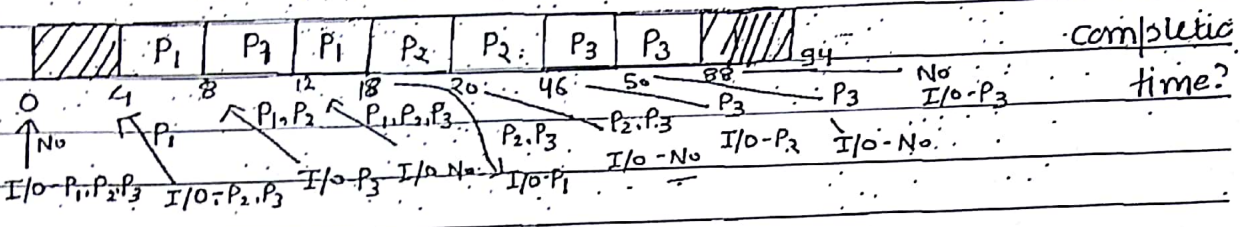
NOTE- 1. The process first spend CPU time followed by I/O time and followed by CPU time again.
2. I/O time of the processes can be overlapped as much as possible.



Date : / /
Page No.

Ques -	P.No	AT	I/O time	CPU time	I/O time	CT = ~~20~~
	1	0	~~40~~	~~14~~ 16	20	20
	2	0	8	~~28~~ 26	4	50
	3	0	12	~~42~~ 38	6	94

What is the completion time?

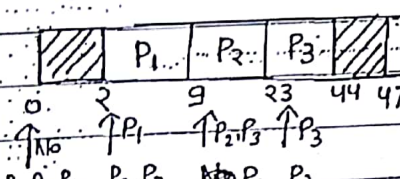


Q8

PNo	AT	I/O	CPU	I/O	CT SRTF
1	0	20	~~70~~	1	10
2	0	4	14	2	~~25~~
3	0	6	21	3	47

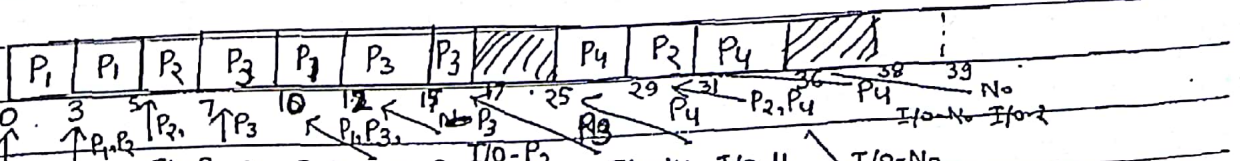
106
47 | 50
47
300

5/47



= 10.6% total time = 47
Idle time = 5
% of being Idle = 5/47 = 0.106
= 10.6%

Ques	PNo	AT	CPU time	I/O time	CPU time	CT
	1	0	~~80~~	~~80~~	2	12
	2	3	20	~~22~~ 18 19 17	2	31
	3	7	~~88~~ 83	0	0	~~38~~ 17
	4	25	~~95~~	2	1	39



Date : / /
Page No.

5. Priority Based Scheduling- Criteria: Priority

Mode: (i) preemptive
(ii) Non-preemptive

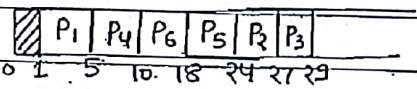
(i) Non-preemptive-

Ques-

Priority	P.No.	AT	BT	CT	TAT	WT
4	1	1	4	5	4	0
3	2	2	3	27	25	22
1	3	3	2	29	26	24
6	4	4	5	10	6	1
5	5	5	6	24	19	13
8 ← High priority	6	6	8	18	12	4

92 64

Avg TAT = 15.33
WT = 10.66



NOTE: if the priorities of the processes are matching, than schedule the process which has lowest arrival time.

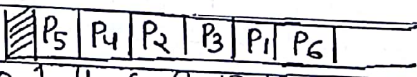
Ques-

priority	P.No	AT	BT	CT	TAT	WT
4	1	4	6	19	15	9
7	2	6	3	21	15	12
6	3	3	4	15	12	8
6	4	2	2	13	11	9
1	5	1	3	4	3	0
3	6	2	2	21	19	17

56 34

Avg TAT = 10.66
Avg WT = 5.66

9
5.66



19 15 9
9 3 0
13 10 6
6 4 2
4 3 0
21 19 17
54 34

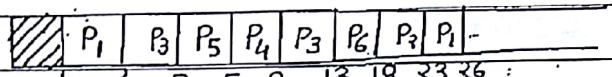
Date : / /
Page No.

Ques

priority	P.No	AT	BT	CT	TAT	WT
3	1	1	3	26	25	21
4	2	2	5	23	21	16
5	3	2	2	13	11	9
6	4	3	3	8	5	2
high ← 7	5	3	10	18	15	5
5	6	4	5	18	14	9

78 59

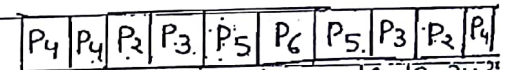
Avg TAT = 13
WT = 8.83



Ques

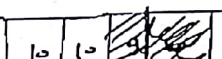
priority	PNo	AT	BT	CT	TAT	WT
2	1	1	8	33	32	24
3	2	2	5	24	22	16
5	3	3	3	19	16	12
2	4	0	2	25	25	22
6	5	4	4	16	12	7
7	6	5	7	12	7	0

114 84



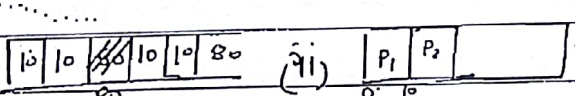
Avg TAT = 19
WT = 13.83

Ques Pg 79 29 - (i)

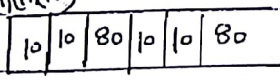


→ 90

(ii)

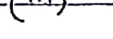


(i)(ii)(iii)

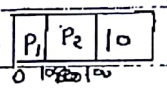


(iv) 5 5 5 5 90

(iii) 10

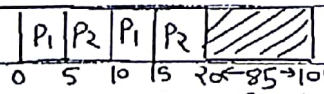


(i)



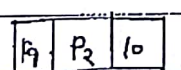
10

(iv)



∴ M - (iv)

(ii)



(ii) 10 90

P1 P2 P3 P4 P5

Date: / /
Page No.

6. Highest Response Ratio Next Scheduling

Criteria: Response Ratio
Mode: Non preemptive

the above algorithm favours the shorter jobs & limits the waiting time of longer jobs.

$$\text{Response Ratio} = \frac{W+S}{S}$$

W - Waiting time
S - Service time / Burst time

- the process with highest response ratio will served first.

Ques-

PNo	AT	BT	CT	WT	TAT
1	0	3	3	0	3
2	2	6	9	1	7
3	4	4	13	5	9
4	6	5	20	9	14
5	8	2	15	5	7
				20	40

Avg TAT & Avg WT = ?

P1	P2	P3	P5	P4
0	3	9	13	15	20

$RR_{P3} = \frac{5+4}{4} = \frac{9}{4}$ $\quad \frac{9}{5}$ $\quad \frac{3}{2}$

$RR_{P4} = \frac{7+5}{5} = 2.4$

$RR_{P4} = \frac{3+5}{5} = \frac{8}{5}$ $\quad RR_{P5} = \frac{2+5}{2} = 3.5$

$RR_{P5} = \frac{1+2}{2} = \frac{3}{2}$

∴ Avg TAT = 8
WT = 4

Ques -

P.No	AT	BT	CT	TAT	
1	1	2	3	2	0
2	4	6	10	6	0
3	5	3	14	9	6
4	6	5	19	13	8
5	8	1	11	3	2
				33	16

Avg TAT = 6.6
WT 3.2

	P1		P2	P5	P3	P4
0	1	3	4	10	11	14	19

$RR_{P3} = 8/3$ $\quad RR_{P3} = 9/3$

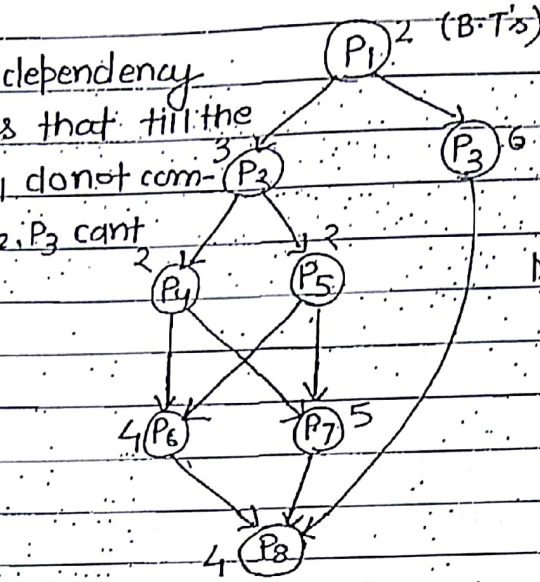
$RR_{P4} = 9/5$ $\quad RR_{P5} = 10/5$

$RR_{P5} = 3/1$

Date: / /
Page No.

Multiprocessor Schedule.

Ques - Consider the following dependancy graph b/w the proces

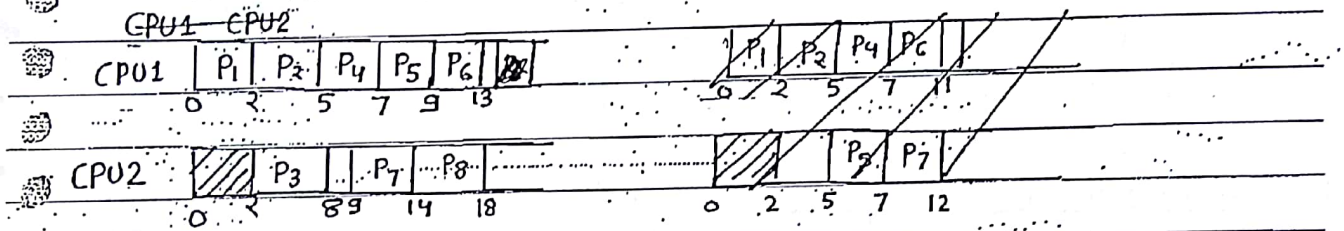


- Here dependency indicates that till the time P_1 donot complete, P_2, P_3 cant start.

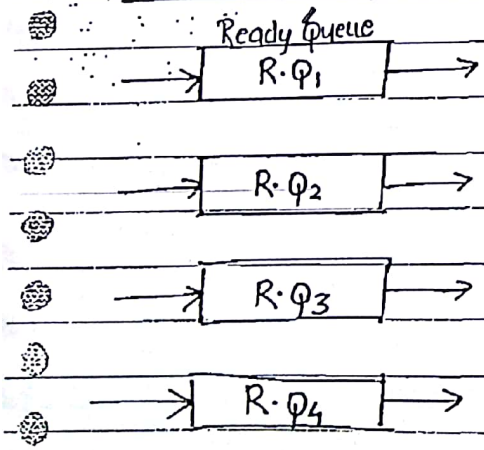
At what time processes complete their exen usin 2 CPU processor ?

Note- 1. One process cannot use the 2 CPU's at same tir
2. Use non-preemptive mo

M -18



7. MultiLevel Queue Scheduling



- the processes get are placed in ready queue on the basis of priority.
- the queue no 1 (R.Q1 high priority processes will be placed in top level Ready Queue.

1. Depending on the priority of the process, in which particular ready queue, the process has to be placed will be decided
2. The low priority process will be placed in bottom level ready queu
3. Only after completion of all the processes from the top level ready

2 2
8 7
12 12
17
19 5

Date: / /
Page No.

4. if this is the strategy followed than the processes which are placed in bottom level ready queue will suffer from ~~st~~ starvation.

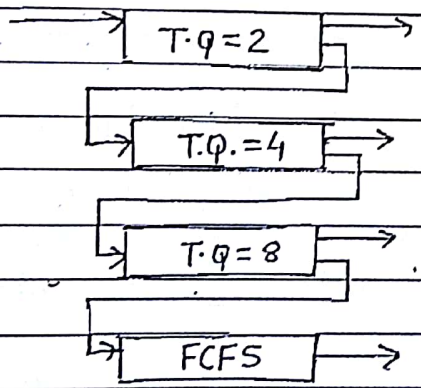
Starvation - The indefinite waiting of a process is called as starvation.

To Avoid problem of starvation, the concept of Ageing is used

Ageing : 1. If the waiting time of a process increases than the priority of the process will be increased.
2. if the age of a process increases by increasing the priority, the process will definitely get a chance to execute and starvation ~~prog~~ problem ~~has~~ will be avoided.

(Age : WT)

8. Multilevel Feedback Queue - T.Q = time quantum



- it also suffer from starvation, to avoid this, we use heuristic scheduling
- if a process in RQ1 it execute for TQ=2 if it complete it get terminated else (interrupted in b/w) it get shifted to RQ2 to execute with TQ=4 & so on continues.

Heuristic Scheduling = Multilevel Feedback + shifting on the Queue basis of ageing.

Ques 8 Pg 77

RQ1 TQ = 2
RQ2 TQ = 7
RQ3 TQ = 12
RQ4 TQ = 17
RQ5 TQ = 23

- Queue = 5th
- No of time it interrupt = 4

Date : / /
Page No.

Ques

PNo	AT	BT	CT	TAT
1	0	2	2	0
2	1	3	5	2
3	2	6	11	5
4	3	5	16	11
5	4	7	23	16
6	5	8	31	23

Throughput - No of job per unit time

Avg TAT = $\frac{57}{6}$ = 19 sec

1 unit 1 ~~ms~~ process TAT = 19 sec

∴ 1 job take = 19 sec

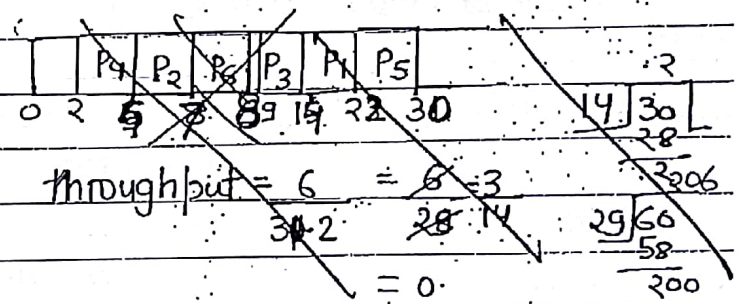
1 sec = $\frac{1}{19}$

Throughput = $\frac{\text{No of process}}{\text{Max CT} - \text{Min AT}} = \frac{6}{31-0}$ = 0.05

$= \frac{6}{31} = 0.19$

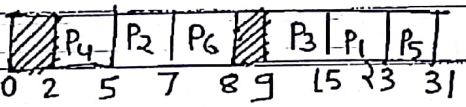
Ques:- Find out throughput using FCFS?

PNo	AT	BT
1	12	8
2	13	2
3	9	6
4	2	3
5	15	8
6	3	1



Throughput = $\frac{6}{30-2} = \frac{6}{28} = 3$

$= 0$



Throughput = $\frac{6}{31-2}$

$= \frac{6}{29}$

$= 0.206$

Schedule length = Maximum CT - Minimum AT

Date: / /
Page No.

Algorithm	Starvation
(i) FCFS	No (there may be some waiting but a definite waiting)
(ii) SJF	Yes
(iii) SRTF	Yes
(iv) RR	No
(v) LJF	Yes
(vi) LRTF	No
(vii) NP Priority	Yes
(viii) preem. priority	Yes
(ix) HRRN	No
(x) M.Q.S	Starvation
(xi) M.F.Q.S	YES
(xii) Heuristic	NO

NOTE: Majority of the operating system practically implement Round Robin Scheduling.

Ques

* CPU scheduling- $\frac{\text{Useful Time spent by CPU}}{\text{Total time spend by CPU}}$

3 cases are there for I/O

I/O
- I/O may occur alone.
- I/O may occur along with CPU.
- multiple I/O's occured together.

Dispatch Latency - time taken to stop one process & start another process.

Concept - 1: In the shortest job first scheduling Algorithm, the burst time of the processes will be expected by using the following formula

$$\{T_{n+1} = \alpha T_n + (1-\alpha)T_n\} \quad 0 \le \alpha \le 1$$

- α - control the relative weight of recent & past history
- T_{n+1} ⇐ next expected burst time
- T_n - previous Actual burst time
- T_n - previous expected burst time

Andrew S. Tannenbaum

Date: / /
Page No.

CPU Synchronization

Processes are categorized into two types-
(i) Cooperative processes.
(ii) Independent Processes.

The execution of one process affects or affected by other process than those processes are said to be cooperative processes, otherwise, they are said to be independent processes.

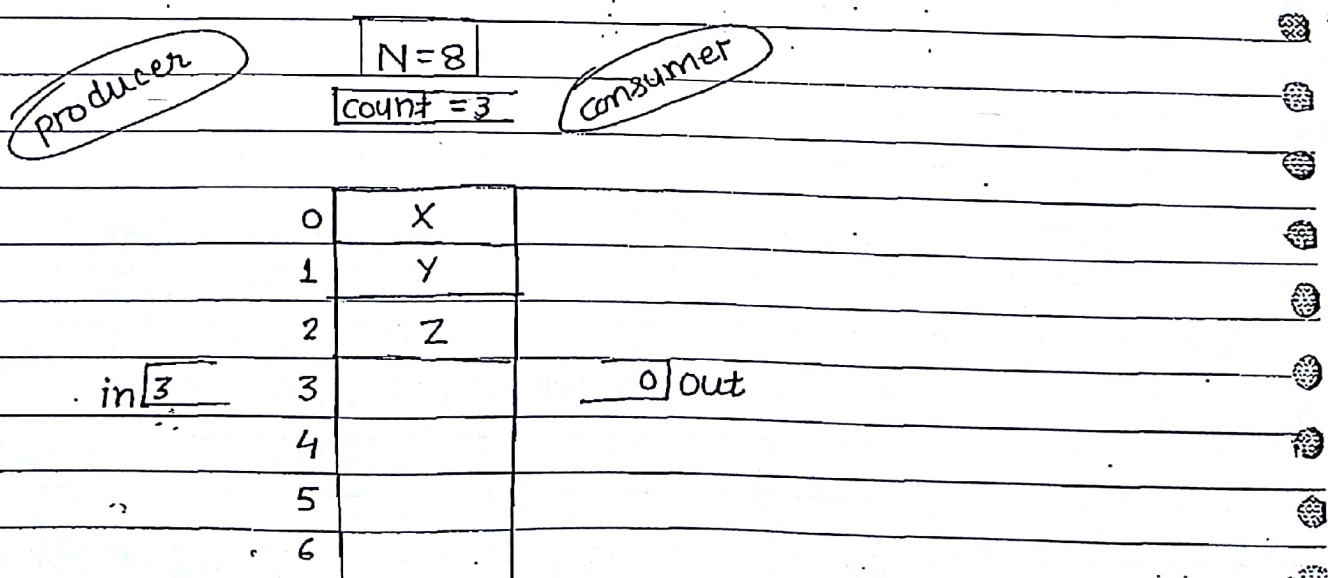
- for independent processes, there is no need of synchronization

Understanding Synchronization-
1) problems which arises due to No synchronization between the processes.
2. Condition to be followed to achieve synchronization.
3) Solutions (Wrong and right).

Step 1- problem due to lack of synchronization-

Producer-Consumer Problem-

~~- in is used by pro~~



Date : / /
Page No.

```
int count=0;
void producer(void)
{
    int itemp;
    while (TRUE)
    {
        producer_item(itemp);
        while (count==N);
        Buffer[in] = itemp;
        in = (in+1) Mod N;
        count = count+1;
    }
}
```

```
1. Load Rp, m[count]
II. INCR Rp
III STORE m[count], Rp
```

```
void consumer(void)
{
    int itemc;
    while (TRUE)
    {
        while (count==0);
        itemc = Buffer[out];
        out = (out+1) mod N;
        count = count-1;
        process_item(itemc);
    }
}
```

```
I. LOAD Rc, M[count]
II. DCR Rc
III. STORE m[count], Rc
```

- 'In' is a variable used by producer to identify the next empty slot in the buffer
- 'out' is a variable used by consumer to identify from where it has to consume the item
- N denotes slots / place in buffer to put item.
- count is a variable used by both producer & consumer to identify No. of items present in the buffer at any point of time.
- Here, shared resources are count & buffer, N.

Two conditions to be followed for providing synchronization-

1. If the buffer is full, producer is not allowed to produce the item into the buffer.
2. If the buffer is empty, consumer is not allowed to consume the item from buffer.

Date : / /
Page No.

Universal Assumption-

1. the running process can get preempted at any point of time after completion of the current instruction.

main()

Analysis: producing an item-

itemp = A

count ≠ N

∴ Buffer[3] = itemp

in = 4

count = 4 I. Rp [3]

II. Rp [4] suppose here preemption occurs.

∴ it goes to consumer process.

Consuming an item-

itemc

count = 3 ≠ 0.

itemc = Buffer[0] = X this will result into invalid state of count bcoz no of item = 3 & count will have 4.

out = 1

I. Rc = [3]

II. Rc = [2]

III. ∴ count = 2

Now again producer process get its chance & execute its 3rd instn

III. count = [4] ⇒ count = 4
Rp

This problem is called as inconsistency problem.

1. Inconsistency- the producer & consumer are not properly synchronized while sharing the common variable count Hence it is leading to the inconsistency.

Printer

Spooler directory is used to store the document when more than user is requesting for printing of data.

out | 0

Directory [SD]

Rj - respective process register

F-N - File Name

all the processes to identify the next
oler directory.

by printer to identify from where it fe

oler directory & in variable.

Let P_1 execute

pqr.doc

I R_1 [3]

II SD[3] → xyz.doc

III R_1 [4] Now it is preempted

Date : / /
Page No.

When P_1 is preempted, P_2 is scheduled,

P_2 - I - R_2 [3]

II - SD[3] ← pqr.doc it overwrite the file written by P_1, xyz.doc

this problem is called as loss of data.

Loss of Data- the processes are not properly synchronized while sharing the common variable "m". Hence, it is leading to the 'loss of Data'.

3 DeadLock:- If the processes are not properly synchronized, while sharing common variables, and common resources it is also possible for deadlock.

Definitions- 1. Critical Section- the portion of program text where shared variables or shared resources will be placed.

Eg: [count = count + 1] in producer-consumer problem.

2. Non Critical Section- the portion of program text where the independent code of the ~~progra~~ processes will be placed.

Eg- in, out in PC problem in = (in + 1) Mod N

3. Race Condition- the final output of any variable depends on execution sequence of the processes. this type of condition is called as Race condition.

Ex- In PC problem if sequence changes to C-I

C-II

P-II

P-II

Date: / /
Page No.

If P → I
P → II
C → I
C → II
C → III
P → III

Here count value is dependent on sequence of exⁿ of instructions.

⇒ count = 4

Note- To avoid the race condition, only one process should be allowed into critical section at any point of time

Step2- Conditions to be followed to achieve synchronization-

1. Mutual Exclusion (M.E) : No two processes may be simultaneously present inside the critical section at any point of time.
2. Only one process is allowed into critical section at any point of time.

2. Progress: 1. No process running outside the critical section should block the other interested process from entering into critical section when critical section is free.
2. if there is only one process trying to enter into critical sectio it should be definitely allowed into critical section.
3. if two or more processes are trying to enter into critical section than one process should be definitely allowed into critical section.

- When two or more process are trying to enter critical section and no one is allowed, than deadlock happen ie no progress.

3. Bounded Wait- 1. No process should have to wait forever to enter into critical section

Date : / /
Page No.

2. there should be a bound on getting chance to enter into critical section.
3. Some processes are indefinitely waiting to enter into critical section, because critical section is always busy by some other processes this situation should not arises.
4. if the bounded waiting is not satisfied, than it is possible for starvation.

4. No Assumption related to hardware & processor's speed.

10/oct/2016

Step3- Solutions- 1. Software Type-
(a) Lock variables.
(b) Strict Alteration or Decker's Algorithm.
(c) Peterson Solution.
2. Hardware Type
Test & Set Lock (TSL) instruction Set.
3. O.S type-
(a) counting Semaphore
(b) Binary Semaphore
4. Programming language compiler support type
(a) Monitors

1. Software Type- (a) Lock variables-

Any No of processes can use this code to access the CS.
R_i - respective process register

Entry Section
I Load R_i, m[lock]
II CMP R_i, #0
III JNZ to Step1
IV Store m[lock] #1
V critical section

lock
0 CS is free
1 CS is busy

- Till the time lock is again made 0 by the process leaving CS, CS is considered to be busy.

Analysis- Lock = 0 [CS]

Let P_1 execute first
$P_1 \rightarrow$ I R_1 [0]
$P_1 \rightarrow$ II CMP $R_1 = 0$
$P_1 \rightarrow$ III execute Let it get preempted here
$P_2 \rightarrow$ I R_2 [0]
$P_2 \rightarrow$ II CMP $R_2 = 0$
$P_2 \rightarrow$ III executed
P_2 - IV Lock = 1
P_2 - V in critical section If preem
P_1 - IV Lock = 1
$P_1 \rightarrow$ V get entered into critical section.

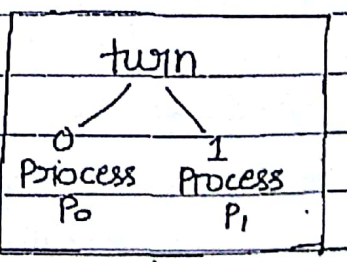
[CS P_2]

[CS P_2 P_1]

- Both processes get entered to the critical section at same time
∴ Mutual Exclusion fails
Hence it is an incorrect solution.
- We have proved that both the processes P_1 & P_2 are entering into critical section at the same time, hence, mutual exclusion is not satisfied & the solution is founded to be incorrect.

(b) Strict Alteration or Decker's Algorithm- Process takes turn to enter into critical sectio

Process Po code	Process P1 code
while(TRUE)	while (TRUE)
{	{
NON_CS();	NON_CS();
while (turn != 0);	while (turn != 1);
CS	CS
turn = 1;	turn 0;
}	}

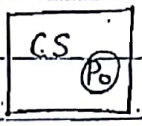


Analysis- Initially turn = 0 & critical section is free

Let P_0 executed first

since turn = 0

it enters to CS



Suppose P_0 get preempted here, P_1 start exec^{n}

since turn = 0 ≠ 1 ∴ it enters to infinite loop

∴ it is not allowed to enter in critical section

When P_1 get preempted. P_0 resume.. than turn = 1 & P_0 leaves CS. Now P_0 can enter to CS.

therefore, Mutual exclusion is satisfied.

Let turn = 0 & CS is empty & P1 is trying to enter into critical section, it is not allowed to enter into CS (turn = 0 ≠ 1), due to entering to infinite loop.

Only one process was trying & not allowed to enter in CS, Hence progress is not satisfied.

• We have proved that progress is not satisfied & the solution is be found to be incorrect.

15. $S_1 = 0$ $S_2 = 0$ – No progress

Mutual exclusion satisfied.

9. $w_0 = F$ $w_2 = F$

8 turn = 0 1

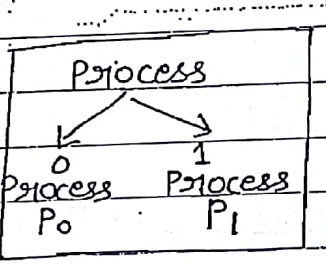
Date : / /
Page No.

Peterson's Algorithm - ∘ two process solution

```
# define N 2 ;
# define TRUE 1 ;
# define FALSE 0;
int turn ;
int interested [N];
void Enter_region (int process)
{
    1. int other;
    2. other = 1 - process;
    3. interested [process] = true;
    4. turn = process;
    5. while ( turn == process && interested [other] == TRUE);
}
```

C-S

```
void Leave_Region (int process)
{
    interested [process] = false;
}
```



initially

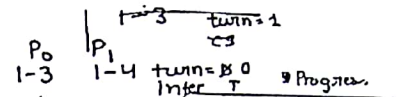
interested [0] = FALSE
interested [1] = FALSE;

∘ turn is a shared variable used by both the processes P_0 & P_1.

∘ till the time interested[~~fals~~process] ≠ false, it is assumed that proces is under CS.

Points- (i) Preemption is just a temporary stop the process will come back & continue the remaining execution.

(ii) If there is any possibility of sol^n becoming wrong by taking preemption, than consider the preemption.



(iii) If any soln is having deadlock, the progress is not satisfied.

To check whether a soln satisfy deadlock or not, first check for deadlock, if there than No progress. if Not there, than check for other condn.

Analysis 1- initially interested[0] = FALSE TRUE
interested[1] = FALSE TRUE
turn = 0 1

Let bring P_0 to CS initially

Process P_0	Process P_1
Process = 0	Process = 1
other = 1-0 = 1	other = 0
5. it is false as int[1] = False ∴ P_0 enter into CS.	5. True (turn=process & true) ∴ infinite loop not allowed to enter into CS.

CS P_0

Let preempted P_0 & P_1 try to enter CS

Analysis 2- Let P_1 start & try to enter into CS
interested[0] = FALSE TRUE
interested[1] = FALSE TRUE
turn = 1

Process P_1	Process P_0
other = 1-1 = 0	other = 1-0 = 1
5. False ∴ it get entered to CS	5. TRUE, it is not allowed to enter in CS

Now after 5th stmt preempted P_1 & start P_0

CS P_1

Hence, Mutual Exclusion is satisfied.

Date: / /
Page No.

2. Progress- $P_0 \rightarrow$ I
II other = 1
III int[0] = True

P_1 I
II other = 0
III int[1] = True
IV turn = 1
P_1 V → turn == process && interested[other] == True ⇒ T

Hence P_1 is not allowed to enter into CS while it is empty.

Progress is not satisfied but here P_0 is only preempted it will continue after sometime & get entered to CS & finally progress will happen

Progress is not satisfied only when it is permanently blocked i.e cannot enter to CS by making some preemption as well

3. Bounded Wait- time is bounded in such a way that you can say that after 'x' no of processes, P_0 will surely get a chance.

Let P_0 get entered to critical section & P_1 is trying to enter in it. When P_0 will exit CS, P_1 will definitely get chance to enter into CS. ∴ Here, it will get a chance to get enter to CS after completion into 1 process.

Ques - We have proved that all the 3 conditions are satisfied in the above solution and the solution is found to be correct. it is properly providing synchronization to processes to access the common Critical section.

Ques - Assume that both the processes P_0 & P_1 are trying to enter in critical section at same time than which process will enter into critical section first?

(a) the process which execute. stmt. 2 first

(b) 3 first

Date : / /
Page No.

(a) 2 first- other=0 Let P_0 execute it first

$P_1 \rightarrow$ III - int[1]=T turn=1

↓ true

3 first : int[0]= true int[1]=T

Turn &= 1

int[1] = true int[0]=true turn=0 → enter

4 [turn=0 turn=1 int[1]=false → No chance

turn 1 → No chance

for statement 3- first do for P_0 & than for P_1

for statement 4 - first do for P_1 & than for P_0 so that difference can be clearly known

$P_0 \rightarrow$ I

II other=1

III int[0]=True

P1 → I

II other=0

III int[1]=True

You have to make strict exc^n alternate as they are competing to get enter to critical section at same time.

P_1 - IV - turn=1

P_0 → IV turn=0

P_0 V - Cond^n become true ∴ it dont get enter to CS

P_1 V - Cond^n become false ∴ it get enter to CS

∴ for statement 4-

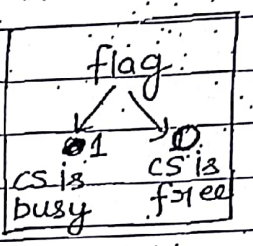
Date: / /
Page No.

Hardware Type-

TSL instruction Set-

TSL Register flag - copies the current value of flag into register & stores the value of '1' into flag. In a single atomic cycle without any preemption.

Entry Section- I. TSL R_i, m[flag]
II. CMP R_i, #0
III. JNZ to Step 1
IV. C.S
V. Store m[flag], 0

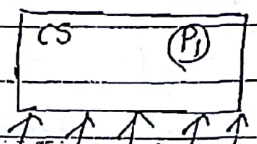


- Any No of process are allowed to access the code.
- Unless & until flag is made 0 after entring in to CS, it is considered that process is under CS.

Analysis. Flag=0 | CS

Let P_1 is trying to enter into CS
P_1 → I R_1 [0] Flag=1
II CMP R_1=0 true
III No Need to jump
IV CS
P_2 → I R_2 [1] flag=1
P_2 II CMP R_1=1≠0 False
P_2 - III JMP to step1

Here Mutual Exclusion is satisfied.
Here progress is also satisfied. bcoz no deadlock exist & every process is getting a chance to enter in CS with preemption
Bounded Wait is' Not satisfied bcoz Analysis- P_1 – I
P_2 – II
P_3 – III
P_1 – IV
P_2



Date: / /
Page No.

When P_1 leaves critical section, one of the process of $P_2 \ldots P_n$ will get chance to enter into CS but we cannot be sure that P_2 will definitely get chance to enter into cs after 'x' process. that's why, it doniot satisfy bounded wait.

NOTE- 1. If some process is in the critical section than all the other processes which are trying to enter into critical section will be repeatedly checking for I, II, III instructions. these processes are busy in checking these instruction & waiting to enter into critical section & they are wasting the CPU cycles (time). this is called as "Busy waiting".

Busy Waiting is also called as "Spin-Lock".

2. To avoid this busy waiting, semaphores will be used.

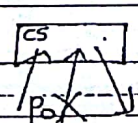
Solution	Mutual Exclusion	Progress	Bounded Waiting
1. Lock variable	X	✓	X
2. Decker's Solution or strict Alteration	✓	X	✓
3. Peterson Solution	✓	✓	✓
4. TSL instruction Set	✓	✓	X

Progress-

Decker's Soln

Process → P_0, P_1

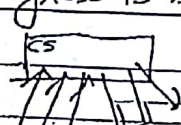
progress is not satisfied



Peterson Soln

Process → P_0, P_1

progress is satisfied



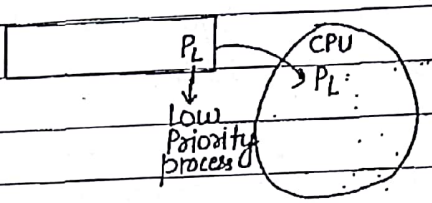
Let P_0 try to enter into CS & get into CS & came out it can do this activity many no. of times

Date: / /
Page No.

In decker soln - Po get enter to cs & came outside. Now you can not send it again than progress is not satisfie

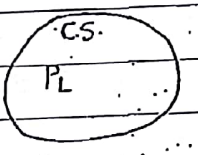
• This concept to check progress must be applied only after checki of deadlock.

Priority inversion problem-



Initially, P_L is alone presen in RQ so it get scheduled t CPU. After some time,

P_L I → R_L = 0
II Flag = 1
III - compare
IV - entered to cs

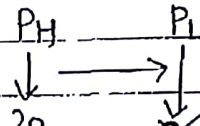


Live Lock
P_L - Ready
P_H - Running

After some time, P_H (high priority process) entered to R.Q than P_L has to be preempted & P_H has to be schedulled to CPU.

the problem arises, since P_H want to enter to CS but P_L still done complete its CS & Flag = 1 ∴ so PH cannot enter to CS so this condition block both the process, this condition is known as Live Lock.

Solution: Priority inheritance - in this, let P_L & P_H be two process P_L will inherit the priority of P_H



Now both have same priority. We will execute P_L first due to less arrival time. ∴ P_L can complete its execution of critical section & P_H can continue afterwar

Semaphore - 1. It is an integer variable which is used by the various process in a mutual exclusive manner to achieve synchronization.

2. It is a technique to achieve synchronization but the improper usage of it will also give wrong results.

Semaphore is categorized into two types -

1. Counting Semaphore - value lies b/w ($-\infty$ to $+\infty$)
2. Binary Semaphore - value ~~lies b/w~~ (0, 1)

The two different operation will be performed on semaphore variable

(i) Down() or wait() or p()

(ii) up() or signal() or v() or release()

Counting Semaphore -

```
Down(semaphore s)
{
    s.value = s.value - 1;
    if( s.value < 0)
    {
        Block the process & place its PCB in the suspended list();
    }
}
Up(Semaphore s)
{
```

Date: / /
Page No.

1. After performing down() op^n, if the process is getting suspended than it is called as unsuccessful down op^n
2. If it is unsuccessful down operation than process will not continue further execution.
3. After performing down op^n, if the process is not getting suspended than it is called as successful down operation.
4. If it is successful down operation than only process will continue further execution.
5. If the initial value of semaphore is +9 than we can perform 9 successful down op^n.
6. the down operation on the counting semaphore will be successful only if the initial value of the semaphore $\boxed{s \geq +1}$
7. There is no unsuccessful up operation, the up operation is always successful.
8. The process performing up operation will definitely continue the execution.

Ques- Consider a system where the initial value of counting semaphore is $s = +17$. the various semaphore operation like 23p, 18v, 16p, 14v, 1P are performed what is the final valu of the counting semaphore?

Ans- $s = +17 - 23 + 18 - 16 + 14 - 1$

$= +17 - 8 = \boxed{9}$ (C)

Note- S=1 down() → S=0 successful

down() S=-1 Unsuccessful blocked=1

down() S=-2 Unsuccessful blocked=2

The negative value of the semaphore (counting S) indicates that no of process suspended or blocked.

operations like 20p, 12V are performed than what is the initial value of semaphore so that one process will be blocked?

Ans- $S = 8 - 20 + 12 = -1$

$\boxed{8 = 7}$

Binary Semaphore

```
Down(Semaphore s)
{
    if(s.value == 1)
        s.value = 0;
    else
    {
        Block the process &
        place its PCB in the
        suspended list();
    }
}
```

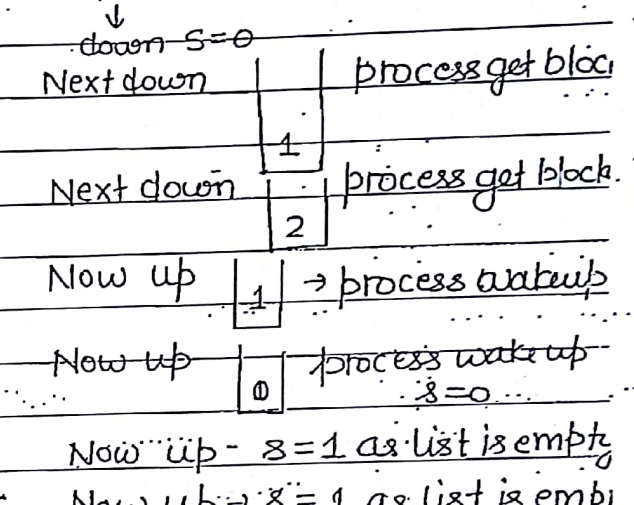
```
Up(Semaphore s)
{
    if(suspended list() is Empty)
        s.value = 1;
    else
    {
        select the process from
        the suspended list &
        wakeup();
    }
}
```

1. After Performing down operation, if the process is getting suspended than it is called as unsuccessful down operation.
2. If it is unsuccessful down op^n, process will not continue its further execution.
3. After performing down operation, if the process is not getting suspended than it is called as successful down operation.
4. If it is successful down op^n than only process will continue its further execution.
5. the down op^n in binary semaphore is successful only if the initial value of semaphore is 1.

Date: / /
Page No.

6. there is no unsuccessful up operation, the up operation is always successful.
7. the process performing up operation will definitely continue the execution.

Note- let initially S = 1 you perform
~~I do~~ P, P, P, V, V, V, V



23 0

if s=1 (initially), n down op^n will made (n-1) process to get suspended
for empty of suspended list with n process → n-s op^n is required but s will remain 0
from n+1 → s=1
for more op^n > n+1 → s=1

Note (Applicable for both Counting & Binary Semaphore)-

1. Every semaphore variable will have its own suspended list.
2. the down & up operations are atomic & no preemption will take place while performing these op^n. (this can be achieved by disabling the interrupt)
3. ~~while perform~~ If more than one process in the suspended list and every time when we perform 1 up op^n, the one process will wake up & this will be based on FCFS.
4. If two or more processes are in suspended list & there is no other process, to wake up these processes than these processes are said to be involved in deadlock.

```
P(mutex);
[CS]
V(mutex);
}
```

process P_{10} execute the following code:

```
while (True)
{
    V(mutex);
    [CS]
    V(mutex);
}
```

what is the max No of process that may present inside the critical section at any point of time.

NOTE: Initial value of Binary semaphore: mutex = 1

(a) 2 (b) 3 (c) 9 (d) 10 (e) 1

Ans- Analysis-

mutex = 1 ... Let P_1 enters to critical section

P_1 - Mutex = 1 → 0

enter to CS.

P_2 tries to get enter to CS but mutex = 0 ∴ get blocked.

P_{10} → V[mutex] → true enters to CS as always successful. but list is not empty so it awake P_2

When P_2 awake, it again get blocked

If P_{10} leave CS → list is empty

∴ mutex = 1

Now P_2 can enter to the CS

mutex = 0

3 ← P_{10} also enter to the CS

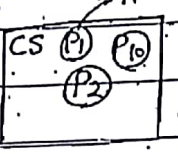
So ans is the answer

Date : / /
Page No.

Ques $P_1 - P_9$ ~~whi~~ same code P_{10} while(1)
{ v(mutex);
CS
p(mutex);

What is max No of processes may present? inside the critical section at any point of time?

(a) 2 (b) 3 (c) 9 (d) 10 (e) 1



When P_1 enters mutex = 0
P_{10} enters up always successful
P_{10} enters & set mutex to 1
P_2 get enters to CS.
if we apply same for P_3, one has to come out (P_1 or P_2) else to put P_3 into critical section & P_{10} came out it get blocked.
So on maximum only 3 process are entertained under CS.

Ques- $P_1 - P_9$ same code P_{10} while(1)
{ p(mutex);
mutex = 1 CS
P_1 enters → mutex = 10 v(mutex);
Now P_{10} tries to enter it }
get blocked.
P_1 leave CS ∴ P_{10} is now activated.
Now if P_{10} enter Mutex = 0 ∴ it again get blocked.
∴ [Maximum No of Process allowed = 1]
this soln is also a correct soln

when code have [p(s) → CS → v(s)] sequence → result to correct soln

Date: / /
Page No.

Ques Consider the concurrent process P & Q executing their respective codes-

process P	Process Q
while(true)	while (true)
{	{
W: ____;	y: ____;
print('0');	print (1);
print("0");	print (1);
X: ____;	Z: ____;
}	}

what must be the binary semaphore op^n on W, X, Y, Z respectively & what must be the initial value of semaphore variable S & T in order to get the output always as ...00110011001.....

(a) W=P(T), X=V(T) Y=P(S) Z=V(S) S=T=1

(b) W=P(T) X=V(T) Y=P(S) Z=V(S) S=1,T=0

(c) W=P(T), X=V(S), Y=P(S) Z=V(T) S=T=1

(d) W=P(T) X=V(S) Y=P(S) Z=V(T) S=0 T=1

Ques- Which of the following will ensure that the output string never contain a substring of the form 01^n0 or 10^n1 where 'n' is odd?

(a) W=P(S), X=V(S), Y=P(T) Z=V(T), S=T=1; No

(b) W=P(S), X=V(T), Y=P(T) Z=V(S) S=T=1; No

(c) W=P(S) X=V(S), Y=P(S), Z=V(S), S=1 Yes

(d) W=V(S), X=V(T), Y=P(S) Z=P(T) S=T=1; 00110001 No

17) Deadlock- both P_1 & P_2 get suspended.

17 (d) (a) P_1, P_2, P_1, P_2 – both blocked

(b) P_1, P_2, P_1, P_2 ____

(c) P_1, P_1, P_2, P_2 – both blocked

(d) ____ No Blocking together.

$P_0 \rightarrow S_0 = 0$ $S_1 = 1$ $S_2 = 1$ 00

$P_1 \rightarrow S_1 = 0$ $S_0 = 1$

$P_2 \rightarrow S_2 = 0$ $S_0 = 1$

Now ~~wait eq~~ P_0 can execute again again 0 get printed

Now it cannot print further.

∴ 0 (9) atleast twice

Case3- $P_1 - P_2 \rightarrow P_0$ 00

$P_1 \rightarrow$ blocked

$P_2 \rightarrow$ blocked ∴ atleast once is not possible.

$P_0 \rightarrow$ true

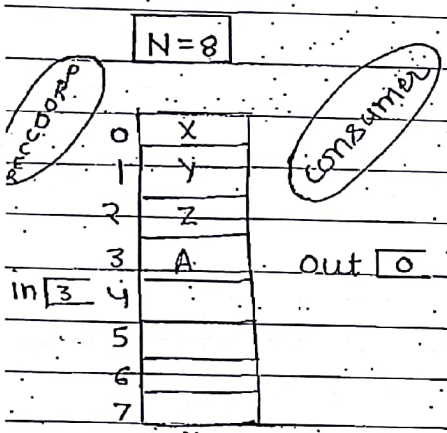
$P_1 \rightarrow S_0 = 1$

$P_2 \rightarrow S_0 = 1$

Date : / /
Page No.

InterProcess Communication-

Producer Consumer with Semaphore-



- mutex is a binary semaphore used by producer/consumer process in mutual exclusive manner, to access the buffer.
- Empty is counting semaphore variable, represents No of slot empty in the buffer at any point of time.
- full is a counting semaphore variable used to represent No of slot full in the buffer at any point of time.

```
Semphore mutex = 1;
Semaphore Empty = N;
Semaphore Full = 0;
Void Producer(void)
{
    int itemp;
    while(TRUE)
    {
1.  produce_item(itemp);
2.  down(Empty);
3.  down(mutex);
4.  Buffer[in] = itemp;
5.  in = (in+1) MOD N;
6.  up(mutex);
7.  up(Full);
    }
}
Void consumer(void)
{
    int itemc;
    while(TRUE)
    {
1   down(Full);
2   down(mutex);
3   Item c = Buffer[out]
4   out = (out+1) mod N;
5   up(mutex);
```

Date : / /
Page No.

Analysis: mutex = 1 Empty + Full = N
Empty = 5
Full = 3

Producer- itemp = A
down(Empty) Empty = 5 4
down(mutex) = mutex = 1 0
in = in + 1 = 4
mutex = 0 1
up(full) = 3 4

Consumer- down(full) ⇒ full = 4 3
down(mutex) → mutex = 1 0
Item c = Bf[0] = X
out = 0 + 1 = 1
up(mutex) ⇒ mutex = 0 1
up(empty) ⇒ empty = 4 5

Here system is consistent as [full + empty = 8]

Case 2- with Preemption- Mutex = 1 0 1 producer- 1, 2,
Empty = 5 4 5 consumer- 1, 2, 3, 4, 5, 6,
Full = 3 2 3 out = 2
item c = 'Y' producer- 3, 4, 5, 6, 7....
itemp = B in ← 5
full = 3

∴ Here also consistency exist.

NOTE- 1. If Buffer is Empty Empty = 8
Full = 0
& down(Full) will block the consumer process, if consumer want to consume.

2. If Buffer is full, Empty = 0 Full = 8
it will block producer process.

Date: / /
Page No.

Ques- What happen if we interchange down(empty), down(mutex) in producer's code?

(a) No problem, the solution still works correct

(b) Both producer & consumer will access the buffer at the same time.

(c) some of the item's produced by the producer will be lost

(d) it is possible for deadlock.

Ans- Producer- down(mutex) ⇒ mutex=0

consumer down(full) ⇒ full = 7 let full=8 (initial)

down(mutex) ⇒ block the process.

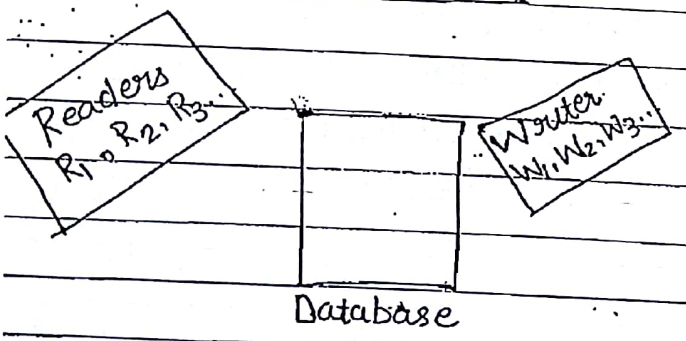
Producer - down(empty) ⇒ block the process let empty=0 (initial)

deadlock occurs.

Ques- What happens if we interchange up(mutex), up(full) in producer's code?

Ans- No problem, it still works correct as now it will (up(full) will also become as part of CS.

Reader-Writer Problem-



- multiple Reader process & multiple writer process & shared common resource is database.
- Reader process reads the database & writer write to it.

rc is integer variable represents reader's count i.e No of readers present in the database at any point of time.

- mutex is binary semaphore used by the readers in a mutual exclusive manner.
- db is also binary semaphore used by both reader & writer in mutual

down(db);

Date: / /
Page No.

```
int rc = 0;
Semaphore mutex = 1;
Semaphore db = 1;
void Reader (void)
{
  while (TRUE)
  {
    down(mutex);
    rc = rc+1;
    if(rc == 1) down(db);
    up(mutex);
    D.B
    down(mutex);
    rc = rc-1;
    if(rc == 0) up(db);
    up(mutex);
  }
}
void writer (void)
{
  while (TRUE)
  {
    down (db);
    D.B
    up(db);
  }
}
```

Four conditions to be followed to provide synchronization-

1. if Reader is reading DB, other writer is not allowed to access the DB.
2. if Reader is reading, other reader is allowed to access the DB.
3. If Reader Writer is in DB, the Reader is not allowed.
4. If Writer is in DB, no other writer is allowed.

Analysis: rc = 0

Case 1- mutex = 1, db = 1 — DB R1

Reader 1- mutex = 1 → 0

rc = 0+1 = 1

if(rc = 1) db = 1 → 0

up(mutex) → mutex = 1

Now it enters to DB.

Let New writer want to enter-

Writer 1- 1, down (db)

b it get blocked so when reader is there, other writer is not allowed to enter.

Let New Reader is enter-

Reader 2- down(mutex) → 1 → 0

rc = 1+1 = 2

rc ≠ 1 ∴ db remain 0

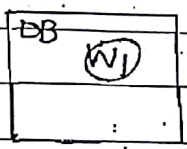
Again mutex = 0 → 1

∴ Reader 2 is also allowed

Date : / /
Page No.

- Here if (rc == 1) down(db); making sure that if one process is present (Reader) in db, then writer are not allowed.
- up(mutex) in Reader process allowed new read to enter ~~wri~~ while a reader is present.

Case II - rc = 0
mutex = 1
db = 1



Let Writer 1 first want to enter -

Writer 1 - down (db) ⇒ db = 0
it enters to DB.

Let New Reader want to enter to DB -

Reader 1 - down(mutex) ⇒ mutex = 0
rc = rc + 1 = 1
if (rc = 1) → true
down (db) ⇒ Reader1 process will be blocked.

∴ When the writer is there in DB than other reader not allowed.

Let New Writer2 want to enter -

Writer 2 - down (db) ⇒ it get suspended.

Ques - What happen if we interchange down(mutex), rc = rc + 1 in the reader code?

(a) No problem still works correct
(b) multiple readers are not allowed in to database.
(c) multiple writers are allowed into database at same time
(d) Both Reader & write coming into db at same time.
(e) it is possible for deadlock.

Date: / /
Page No.

Ans- Let Reader1- rc = 1
mutex = 0
Let writer1- down(db) - 0 → DB W1
DB
Let Reader2- rc = 2
Let Reader 1- if (rc == 1) → false
up(mutex) → mutex = 1
& allowed to database

DB W1
R1

∴ both will be allowed to enter to database at same time.

Ques- What happen if we interchange if condⁿ & up(mutex) in reader's code?

R1- rc = 1
mutex = 0
mutex = 1
R2 ~~rc~~ mutex = 0
rc = 2
mutex 1
W1 → allowed db = 0
R2 ⇒ enter in db
R1 ⇒ enter in db

∴ Both readers & writers are allowed into db at same time.

Ques- What happens if we interchange, ~~rc~~ = down(mutex), rc = rc - 1 in reader's code?

R1 → ~~mut~~ rc = 1 m = 0 db = 0 m = 1
rc = 0 ~~rc = 1 → allowed~~

R1
R2

R2 → rc = 1 → not allowed

R1 → rc = 1 m = 1 db = 0
R2 → rc = 2 m = 1 db = 0

R2 → rc = 2 → (R1 but when rc

Date : / /
Page No.

$R_1 \rightarrow m = \not 0$ rc=1 db=0 m=1
$W_2 \rightarrow$ blocked
$R_1 \rightarrow$ rc=0 ~~mutex=0~~
$R_2 \rightarrow$ rc=1 m=0 rc=1 down(db) → blocked
$R_1 \rightarrow$ ~~mutex~~ down(mutex) ⇒ blocked
Hence, Condition for deadlock.

Ques- 19 R_1 - m=0 R=1 0 1
R≥1 or w==1
(d) w=1
up(mutex) 1

Q0 W1 → W=0 R=0 mutex=0 Q0. d
1
1 R=R+1. down same time.
Writer 1- W=0 R=0 mutex=1
mutex = ∅0.
if(W==1 or R≥1) → false
mutex = ∅1
Reader 1- mutex=0
R=1 mutex=1
enter to database
Writer2- W=1
enter to database
∴ both enter to database at same time.

19. (d) (c) $R_1 \rightarrow$ allowed mutex=1 Q d
$W_1 \rightarrow$ Not allowed
$R_1 \rightarrow$ mutex = 0
$W_2 \rightarrow$ blocked

Date : / /
Page No.

M=1 R=0 W=0

19) (c) R_1 → M=0
R=1
M=1 → enter to db
W_1 → mutex=0
mutex=1
R_1 → mutex=0
R=0
mutex=1
W_1 → mutex=0
allowed w=1 mutex=1 → enter to database
R_2 → mutex=0
w=1
if I applied down(mutex) here it get suspended
R_3 → suspended
R_4 → all suspended
W_2 → down(mutex) → it also get suspended.
Hence deadlock may exist.

∴ we use at ② up(mutex)
① up mutex)

21. S=1 t=1 (b)

Ques- Consider the following code used by classical reader's & writer. which of the following is true

(a) No problem in the code, the sol^n is properly synchronizing classical reader's & writer.
(b) Multiple writers are allowed into database at the same time
(c) Both the reader & writer are coming into DB at same time.
(d) It is possible for deadlock.

Date : / /
Page No.

```
int rc = 0
int wc = 0
Semphore mutex = 1
         db = 1
         r = 1
void Reader (void)
{
  while (TRUE)
  {
    down(r);
    down (mutex);
    up(r);
    rc = rc + 1;
    If (rc == 1) down(db);
    if (wc == 1)
    {
      down(r);
      up(db);
    }
    up(mutex);
    DB
    down mutex;
    rc = rc - 1;
    if (rc == 0) up (db);
    up(mutex);
  }
}
```

```
void writer (void)
{
  while (TRUE)
  {
    down(r);
    wc = wc + 1;
    down(db);
    D.B
    up(db);
    wc = wc - 1
    up(r);
  }
}
```

W1 → r = 1 db = 1 m = 1 rc = 0 wc = 0
r = 0
wc = 1
db = 0

W1

R1 → r → Not allowed

R1 → r = 0 m = 0 r = 1
rc = 1 db = 0
r = block wc = 0

R1 → r = 0 m = 0 r = 1
rc = 1
r = 0 wc = 1 db = 0
r = 0

db

db = 1
r = 0
r = 0
up

$wc=0$ $m=0$

$R_1 \rightarrow rc=1$ $r=1$ $m=0$

$R_2 \rightarrow$ Not allowed

$W_1 \rightarrow db=0$ allowed

$R_2 \rightarrow$ block

$W_2 \rightarrow$ blocked

R-W

W-

$W_1 \rightarrow r=0$ $wc=1$

$R_1 \rightarrow r=0$ $m=0$ $W_1 \rightarrow$ block

$W_1 \rightarrow r=0$ $wc=1$ $db=0$

$db=1$

OB

up(r)

$r=0$ $m=0$ $r=1$ $rc=2$

0 $wc=1$

block

$r=0$ ↑

$R_1 \rightarrow r=0$ $m=0$ $r=1$ $rc=1$

$W_1 \rightarrow r=1$ $wc=1$

$W_2 \rightarrow R_2 \rightarrow r=0$ ✗ block

$W_1 \rightarrow r=0$ $wc=1$

$R_1 \rightarrow db=0$

$W_1 \rightarrow$ blocked

$R_1 \rightarrow$ blocked

Hence both are block therefore deadlock happen

$R_1 \rightarrow r=0$ mutex$=0$ $r=1$ $rc=1$

$W_1 - r=0$ $wc=1$

$R_1 \rightarrow rc=1$ $db=0$

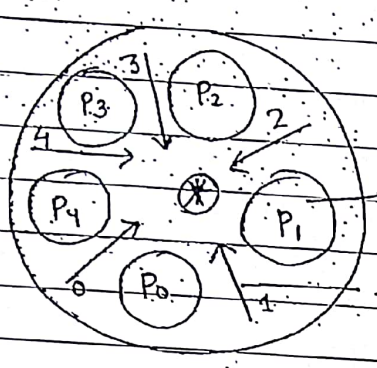
$W_1 \rightarrow$ blocked

$R_1 \rightarrow$ down$r \rightarrow$ blocked

Date: / /
Page No.

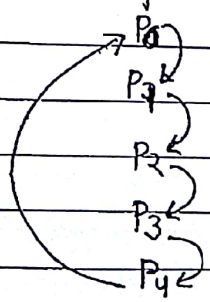
Dining Philosopher's Problem-

N = 5




```
Void philosopher (int i)
{
    while (TRUE)
    {
        Thinking ();
        take_fork (i); // taking left fork
        take_fork (i+1 % N); // taking right fork
        eat();
        put_fork (i); // put left fork back
        put_fork ((i+1) % N); // put right fork back
    }
}
```

- Initially philosophers is in thinking state.
- ~~Only~~
- i is the No. of philosopher.
- chopstick is acting as shared resource & philosopher is the process.
- the problem with this code is each philosopher is taking left fork first, if all the philosopher are hungry at the same time than everyone will take their left fork first & No one get right fork, when they try for it than, all the philosopher will wait on each other for the right fork. they will go in to deadlock



Date : / /
Page No.

```
# define N 5
# define LEFT (i+N-1)%N        // Left philosopher
# define RIGHT (i+1)%N         // Right philosopher
# define THINKING 0
# define HUNGRY 1
# define EATING 2
Semaphore mutex = 1
Semaphore S[N]; // all are S[i]'s are initialized to 0;
int state[N]   // an array to keep track of every one's state.
void philosopher(int i)
{
    while(TRUE)
    {
        Thinking();
        take_forks(i);
        eat();
        put_forks(i);
    }
}
take_forks(int i)
{
    down(mutex);
    state[i] = Hungry;
    test(i);
    up(mutex);
    down(S[i]);
}
put_forks(int i)
{
    down(mutex);
    State[i] = Thinking;
```

i - philosopher No:

- Mutex is a Binary Semaphore used by the philosopher in a mutual exclusive manner.
- S[N] is an array of binary semaphore initially all are assigned to 0.
- State[N] is an integer array used to keep track of every philosopher state.

```
void test (int i)
{
  if (state[i] == HUNGIRY && STATE [LEFT] != EATING && STATE [RIGHT]
      != EATING)
  {
    state[i] = EATING;
    up(S[i]);
  }
}
```

Analysis : mutex = 1

P_0 = T S[0] = 0
P_1 = T S[1] = 0
P_2 = THE S[2] = 1
P_3 = T S[3] = 0
P_4 = T S[4] = 0

} initially

Let P_2 is going to eat - i = 2

take-forks (2) → mutex = 0

state[2] = P_2 = H

test (2)

{ true

S[2] = 1

& P_2 = HE

∴ P_2 go to eating state.

Date : / /
Page No.

Ques Assume that philosopher P_1 & P_3 are in the eating state than the philosopher P_2 is also trying to go into eating state than which statement in the above code is controlling the philosopher not to go into eating state

(a) down(mutex)
(b) If (condition) — bcoz P_2 - 1 = 2
(c) test function
(d) down(S[i]);

take fork(2)
↓ mutex = 0
test P_2 - H
test(2) → false
return
mutex = 1
down(S[i]) → it will bloc the process.

Ques - What happens if we interchange up(mutex), down[S[i] in the take-fork function?

(a) No problem, the soln still works correct.
(b) More than two philosopher will go into eating state.
(c) it is possible for deadlock.
(d) None of the above.

N - possible for deadlock. Let P_4 & P_3 in eating
P_2 tries get blocked
P_1 → put fork - get blocked
P_3 → put fork - get blocked
P_4 → take fork - get blocked
P_5 - take fork - get blocked

Date : / /
Page No.

Question- Let P[0] .... P[4] be processes and m[0] ... m[4] be binary semaphore mutexes all are initialized to 1. Each process Pi execute the following code-

```
wait(m[i]);
wait(m[(i+1) mod 4])
CS
signal(m[i]);
signal (m[(i+1)mod 4]);
```

Consider the below statements

I. Mutual exclusion is satisfied

II. _____ is not satisfied.

III. It is possible for deadlock.

∴ Mutual Exclusion is not satisfied & deadlock can occur

P0 → CS 0 X0
P1 → CS 1 X0
P2 → CS 2 1
P3 → CS 3 1

Not satisfied ME

P0 → I 0 0
P1 → I 1 0
P2 → I 2 0
P3 → I 3 0

P0 → II - blocked
P1 → II - blocked
P2 → II - blocked
block

∴ deadlock occured

Question- Consider the following code used by the processes to access the common critical section

```
Shared Boolean Lock = False;   // global
Boolean Key ;                  // local variable
do
{
    key = TRUE;
    while (key == TRUE)
    swap(lock, key);
[CS]
```

Note- swap fn atomically swap two values using call be refer

Consider stmt-

I ME satisfied

II ME not

Date : / /
Page No.

P_1 - L=0 K=1
K=0 L=1 | CS P_1

P_2 - L=1 key=0 → Can Never enter to critical section.

PLC Support Type

(i) Monitors - 1. Monitor is programming language compiler support type of soln to achieve synchronization.

2. Monitors are collection of variables, condition variables and procedures combined together in a special kind of module or the package.

3. the processes running outside the monitor cannot directly access the internal variable on the monitor but however they can call procedures of the monitors

4. Monitors has an important property that only one process can be active inside the monitor at any point of time.

Ex - Java support implementation of monitor.

Syntax of Monitors -

```
monitor Example          • monitor-keyword
{                        //Example is name of monitor
    variables;
    condition variables;
    Procedure P1
    {
        ≡
    }
    Procedure P2
```

Declaration: condition x, y;

the two different operations will be performed on cond^n variables of the monitor-

(i) Wait operation (wait());
(ii) signal();

wait(); x.wait(); or wait(x); We are performing wait on x.

the process performing wait operation on any condition variable will be suspended and the suspended process will be placed in Block Queue of respective condition variable

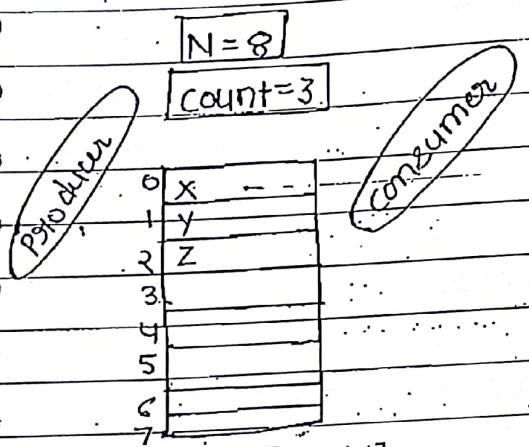
signal(); x.signal(); or signal(x);

signal()

Block Queue is Empty	Block Queue is Not Empty
the signal has no effect & signal will be lost.	one process will be resumed from the block Queue & any one process will continue the execution

Date : / /
Page No.

Producer- Consumer Problem-



- no chance of incons

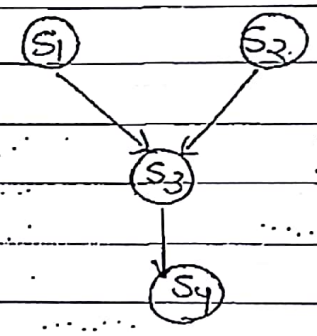
```
monitor producer_consumer
{
int count = 0;
condition Empty, Full;
procedure Enter_item
{
if (count == N) wait (Full);
enter (item);
count = count + 1;
if (count == 1) signal (Empty);
}
procedure Remove_item
{
if (count == 0) wait (empty);
remove (item);
count = count - 1;
if (count == N-1) signal (full);
}
}
produ procedure producer
{
while (TRUE)
{
produce_item (itemp);
producer_consumer.Enter_Item;
}
}
procedure consumer
{
while (TRUE) {
producer_consumer.remove_item;
```

be any inconsistency while updating the count variable because inside the monitor only one process can be active at any point of time.

12-oct-2016- Concurrent Programming

S_1 : a = b + c;
S_2 : d = e * f;
S_3 : g = a / d;
S_4 : h = g * i;

Read-Set = {b, c, e, f, a, d, g, i}
Write-Set = {a, d, g, h}



→ Precedence graph b/w the statements

1. Any two statements S_i & S_j can be executed concurrently or parallely if they are following the below cond^n

(i) $R(S_i) \cap W(S_j) = \phi$
(ii) $W(S_i) \cap W(S_j) = \phi$
(iii) $W(S_i) \cap R(S_j) = \phi$

Concurrent-

→ they donot have any dependency.
→ they can execute concurrently or parallely
→ Anyone can start first. (for single CPU system).

Real concurrent programming is possible only one real system.

Date : / /
Page No.

The concurrent programming will be written by using

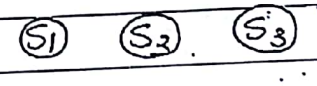
par begin - par end
or
co begin - co end

par - parallel
co - concurrent

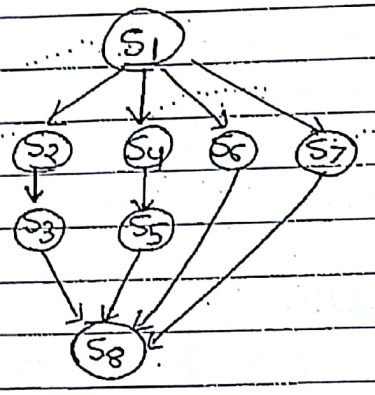
```
begin
  S1;
  S2;
  S3;
end
```


```
par begin
  S1;
  S2;
  S3;
par end
```

par-begin all statements will execute parallely

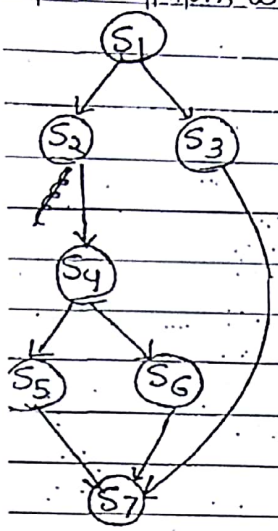



```
program:- S1;
          par begin
            begin
              S2;
              S3;
            end
            begin
              S4;
              S5;
            end
            S6;
            S7;
          par end
          S8;
```



Date : / /
Page No.

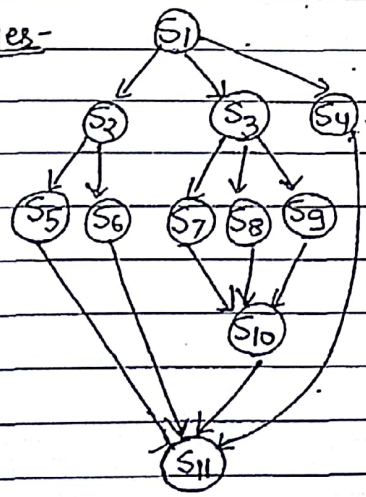
Given Graph, write equivalent concurrent program




```
begin
  S1;
  parbegin
    S2;
    S3;
```

```
begin
  S1;
  parbegin
    begin
      S2;
      S4;
      parbegin
        S5;
        S6;
      parend
    end
    S3;
  parend;
  S7;
end
```

Ques-

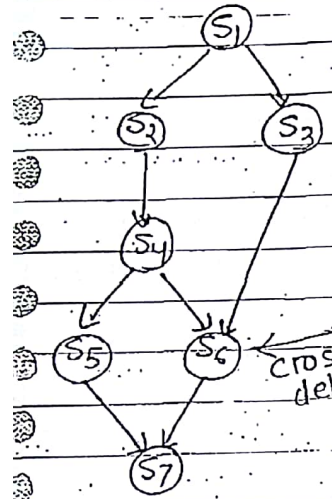



```
begin
  S1;
  parbegin
    begin
      S2;
      parbegin
        S5;
        S6;
      parend
    end
    begin
      S3;
      parbegin
        S7;
        S8;
```

```
      S10;
    end
    S4;
  parend;
  S11;
end
```

Date : / /
Page No.

NOTE- It is not possible to write the concurrent program for all the dependency (Predence) graph by using par begin & par end but it is very much possible with the help of semaphore operations




```
S1
parbegin
begin
S2;
S4;
parbegin
```

→ Not possible to write by parbegin a

This type of dependency is known as cross dependency.

```
parbegin
    begin S1, parbegin V(a), V(b), parend, end;
    begin P(a), S2, V(c), end;
    begin P(b), S3, V(d), end;
    begin P(c), S4, parbegin V(e), V(f), parend, end;
    begin P(e), S5, V(g), end;
    begin P(d), P(f), S6, V(h), end;
    begin P(g), P(h), S7, end;
parend;
```

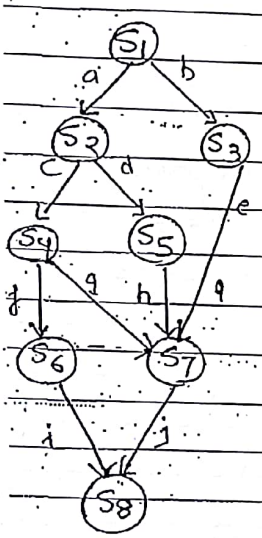
All the binary semaphore a,b,c,d,e,f,g,h are initialized to 0.

Analysis- Since all the stmts are written inside parbegin so they can execute all together. the sequence of exen getting controlled by semaphore.

Date : / /
Page No.

S_2 & S_3 must be executed after S_1 only so suppose the cond^n when S_2 tries to execute first $p(a)$ will suspend it. When S_1 completed, it do $v(a), v(b)$ ie $a=1$ & $b=1$ so that S_2 & S_3 can execute & so on.

Ques:- Write equivalent concurrent program




```
parbegin
  begin S1, parbegin V(a), V(b), parend, end;
  begin p(a), S2, parbegin V(c), V(d), parend, end;
  begin p(b), S3, parbegin V(e), end;
  begin p(c), S4, parbegin V(f), V(g), parend, end
  begin p(d), S5, V(h), end;
  begin p(f), S6, V(i), end;
  begin p(e), p(g), p(h), S7, V(j), end;
  begin p(i), p(j), S8, end;
parend
```

Ques Consider the following concurrent program-

```
int x=0, y=0;
parbegin
  begin
    x=1;
    y=y+x;
  end
  begin
    y=?;
    x=x+3;
  end
```

What can be the final values of x & y after completion of the program?

I. x=1, y=2
II. x=1 y=3
III. x=4 y=6
IV ~~none of the above~~

Which of the claim are possible?

Date : / /
Page No.

1.1
2,3

x=0 y=3 4,3

i) x=4 y=2 4,2

ii) 3 2 3 1,3

iii) 1 2 6 4,6

iv) 3 2 1,3

only II & III are possible.
In parbegin, we can swap the order of execution.

5. 5 4 5 1,5

1 4 10 6,10

6 1 6,5

only I, II, III

25 x=0 S=2

W, X	Y, Z	W → S = 1	Rx = 1 Ry = -2
Read	P	X → S = 0	Rw = 2 Rz = -2
INCR	Read	X = 1 Sel	x = -2
STORE	INCR DECR	Y → S = 0 X = 0 = 2	Ry = -2
V	Store	W 1 1	∴ X = 2
	V	Ry = -2	
Sequence –	Ry = 0	Rz = -2	
Rx = 0	Ry = -2	Ry = -2	
Rx = 1	Rz = 0	Rz = -2	
X = 1	Rz = +2	INCR = 1	
Rw = 1	X ∈ Ry = -2	incR = 0	
Rw = 2	X ∈ Ry = -2		
W = 2			

Rx = 0	x = -2 Ry
Ry = 0	x = -2 Rz
Rz = 0	x = 1 Rx
Ry = -2	Rw = 1
Rz = -2	

Date : / /
Page No.

W	X	Y	Z
P(S)	P(S)	P(S)	P(S)
I. Load R_W, M[X]	I. Load R_x, M[X]	I. Load R_y, M[X]	I Load R_z, M[X]
II INCR R_W	II. INCR R_x	II DECR R_y	II DECR R_z
III STORE M[X], R_W	III. STORE M[X], R_x	III STORE M[X], R_y	III. STORE M[X], R_z
V(S);	V(S)	V(S)	V(S)

Analysis- X=0 S=2

W - I [0] R_W S=1

II [1] R_W

Y - S=0

II [0] R_y

II [-2] R_y

III X = -2

S=1

Z - S=0

I [-2] R_y

II [-4] R_y $\Rightarrow$ X = 2

III X=-4

S=1

W III X=[1]

S=2

X - S=1

I [1] R_x

II [2] R_x

III [2] X

S=2

Date : / /
Page No.

Ques- Consider the following concurrent program

```
int count = 0;
void tally()
{
    int i;
    for (i = 1; i <= 5; i++)
        count = count + 1;
}
main()
{
    parbegin
        tally();   // Let P1
        tally();   // Let P2
    parend
}
```

What is the minimum final value of count after completion of both the function of tally()?

NOTE- count = count + 1 will execute in 3 instructions like

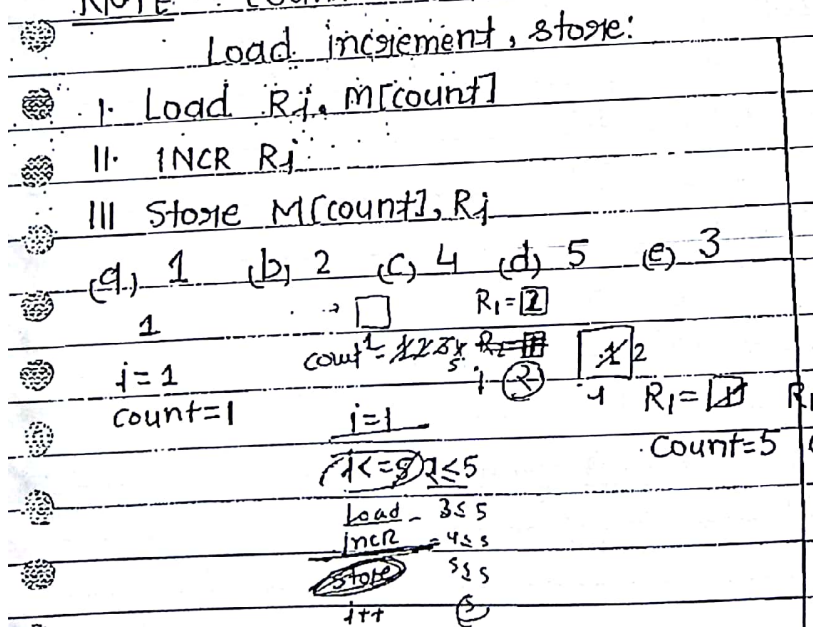
Load increment, store:

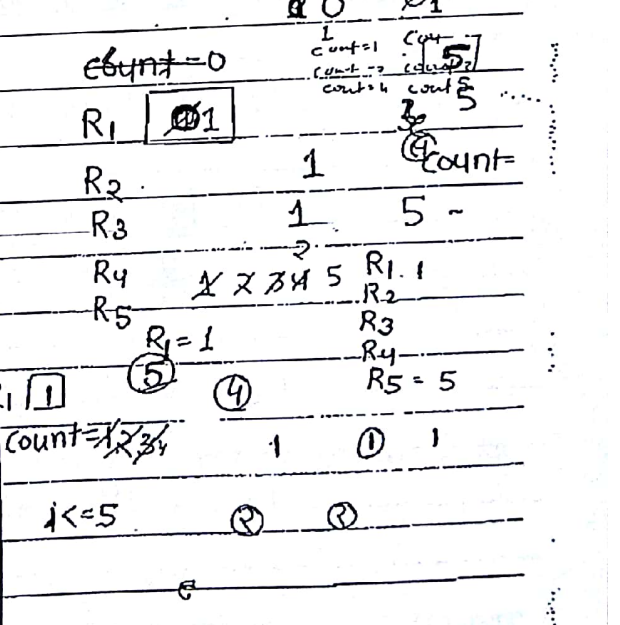
I. Load Ri, M[count]

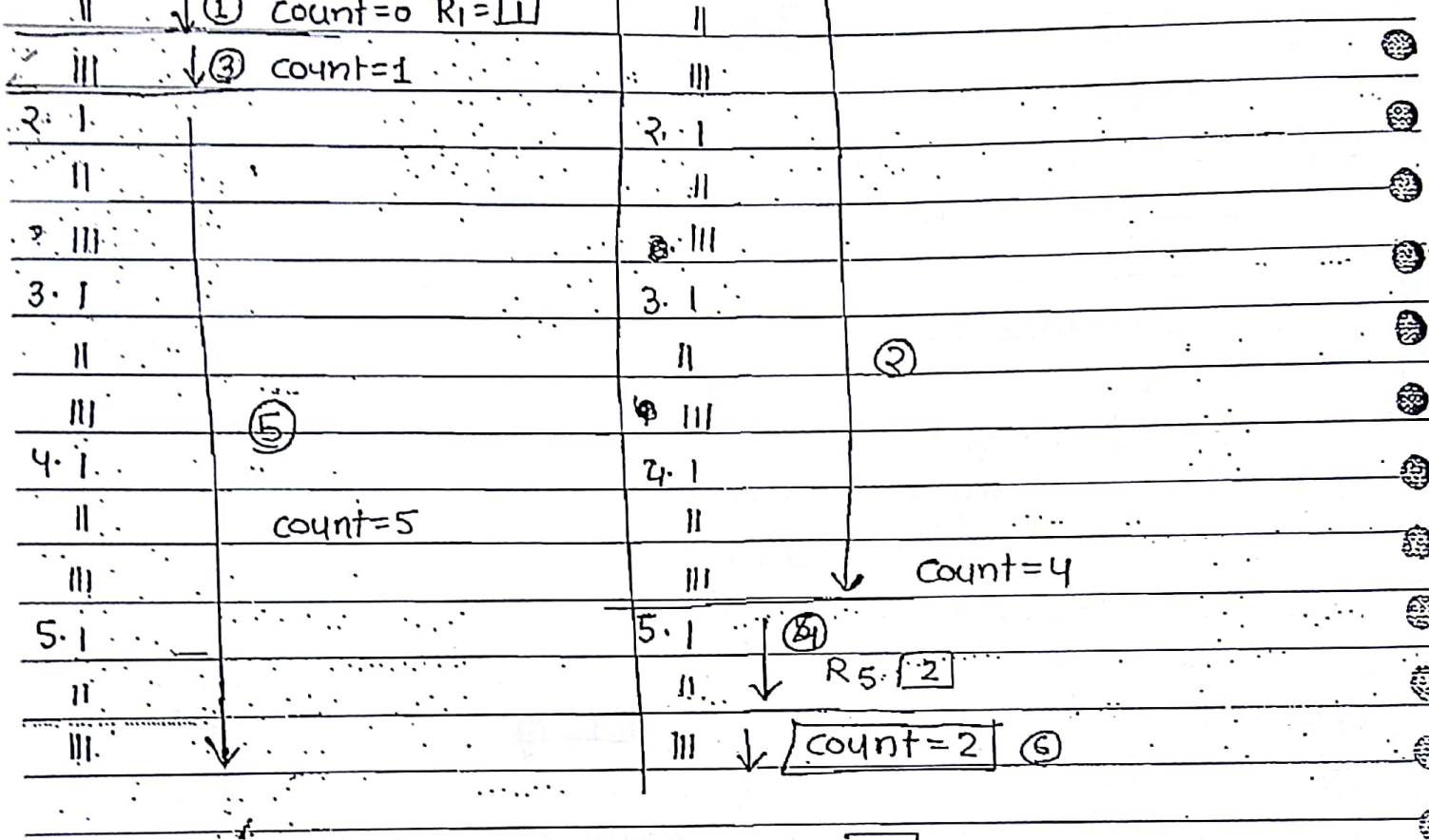
II. INCR Ri

III Store M[count], Ri

(a) 1 (b) 2 (c) 4 (d) 5 (e) 3







Analysis- P_1 - I → ist iteration I R_1 = 0

II INCR 1 count=0

P_2 I, II, III, IV (4 iteration)

count=4

P_1 1st iteration III count=1

P_2 V iteration - I R_5 1

II INCR 2

P_1 → II, III, IV, V iteration (all)

count = 5

P_2 V iteration - III 2

count

Date : / /
Page No.

DeadLock

The two or more processes are waiting on some event to happen, which never happens than those processes are said to be involved in deadlock.

Basics of DeadLock-

Processes- (P1) (P2)

Resources → [R1] [R2] [R3]

Resources with instances-

[R1] [R1] → single instance of Resource R1

[R2 ••] → two instances of resource R2

Requesting Edge- (P1) → [R1]

process P1 is requesting for one instance of resource R1.

- (P1) → [R1] → process P1 is requesting for two instances of resource R1

Allocation Edge-

- [R1 •] → (P1)

One instance of resource R1 is allocated to process P1.

- [R2 ••] → (P2)

two instances of resource R2 are allocated to the process P2.

the resource request & resource allocation will be represented

Date : / /
Page No.

Resource Allocation Graph - is a set of vertices & Edges

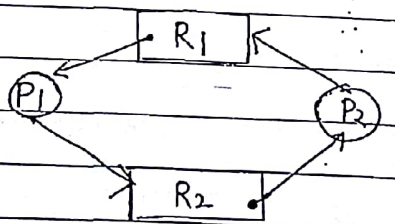
$G = (V, E)$

V - contains Resources & Processes

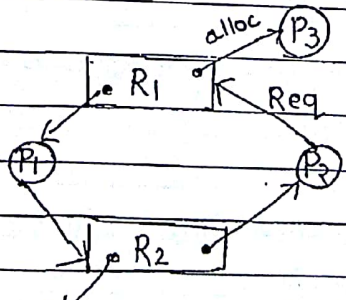
E - Allocation Edges & Requesting Edge

The resource request/release life cycle -

1.) the process will make request for the resource.
2.) the Operating System clearly validates the request of the process i.e. whether the request made by system is valid.
3.) If the request made by process is valid, OS will check for availability of resource.
4.) If the resource is free & available, it will be allocated to the process, otherwise, the process has to wait.
5.) If all the resources required for the execution are allocated, than the process will go into execution.
6.) once the execution of the process is completed, it will release all the resources.



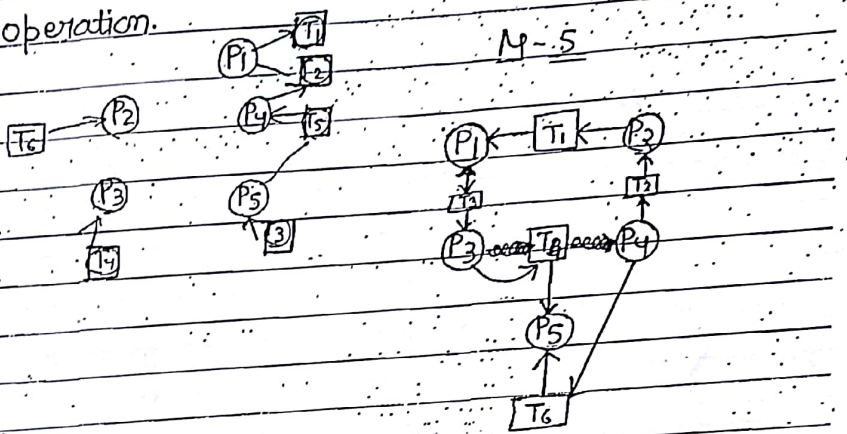
Deadlock exist in R.A. Graph



- No deadlock exist as P_3 has R_1 & require only that. P_3 will complete its execution & P_2 will get the resource & all processes will complete.

Date : / /
Page No.

Ques- Consider the system which has n processes & 6 tapedrives (Resource). Each process requires two tape drives to complete their execution then what is maximum value of n which ensure deadlock free operation.



N = 5

Ques- Consider a system which has 3 processes & each process requires two resources to complete their execution than what is the min. No of resources required to ensure deadlock free execution?

N = 4

Ques- Consider a system which has n processes, 6 resources. Each process requires 3 resources to ~~resource~~ complete the execution than what is the max. value of n which ensure deadlock free execution?

↳ [consider for all cases]

N = 2 B 2 n = 2

Ques- Consider a system which has 3 processes P_1, P_2 & P_3. the peak demand of each process is (5, 9, 13) respectively for the resource R. What is min. no. of resource required to ensure deadlock free execution.

P_1 ↓4 P_2 ↓8 P_3 ↓12 25

N = 25

Date : / /
Page No.

Page 96 Q22- P_1 P_2 P_3 P_4

4 3 7 x11

10

3 2 11 3 2 6 (9)

(10) 20

(9)

M-9

for deadlock-

$3+2+5+x-1 \geq 20$

$x-1 \geq 9$

$x \geq 10$ $x = 11$

for deadlock (10) 9

$x \geq 10$

$\boxed{x = 9}$

II

19. $R = 9 \Rightarrow 4$

20

Note

- Necessary Condition- the deadlock may be possible or may not be possible.....
- Sufficient Condition- the deadlock never be possible.
  - It is the least upper bound which satisfies the condition, is called as sufficient condition.

Ques- Pg- 96 Q13- Resource = m

(c) $\left[\sum_{i=1}^{n} S_i < m+n\right]$

proof- for Not to be cond^n of deadlock-

$S_1 - 1 + S_2 - 1 + S_3 - 1 + S_4 - 1 + \dots + S_n - 1 < m$

$S_1 + S_2 + S_3 + \dots + S_n - n < m$

$\sum_{i=1}^{n} S_i < m+n$

→ proved

14- $k=9$ $n=9$ Req → Res $k=3$ $n=3$

P_0 → R_0

Date : / /
Page No.

~~$P_0 \to ? \quad P_1 \to ? \quad P_2 \to ? \cdots P_{k-1}$~~

$n = 20 \quad k = 10$

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
2	2	2	2	2	2	2	2	2	2
R_0	R_0	R_3	P_6						
R_2	R_6	R_4	k						

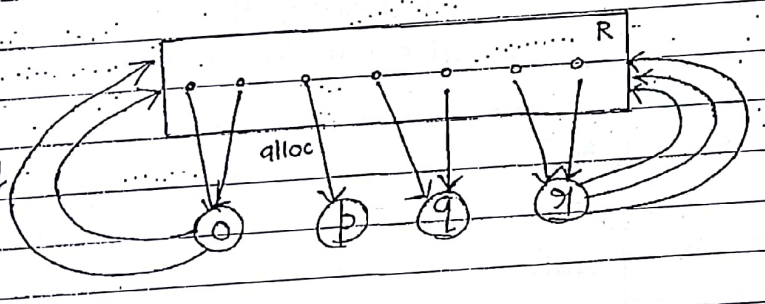
x_1, x_2, x_3

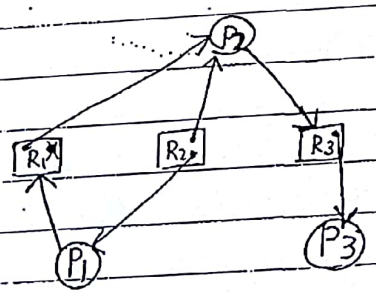
gyog Q10 - P_1 P_2 P_3 No x_p, x_q

y_1 y_2 y_3 $x_p + x_q \geq \max y_k$ $\therefore M - x_p + x_q \geq \max y_k$

y_p } $= 0$ implies that ~~they donot require instance y_p & y_q~~
y_q

$\therefore$ when x_p & x_q complete it must be greater than max y_k so that if any process get y_k, it should complete its execution.





gy 6 (a)

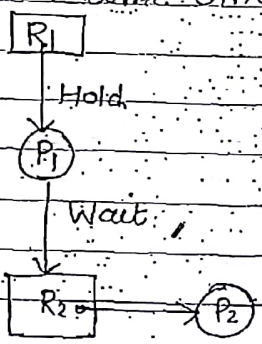
Date: / /
Page No.

Concepts of deadlocks-

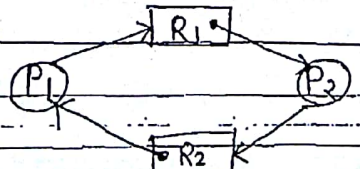
1. Deadlock characteristics.
2. Deadlock prevention. It is tried that cond^n must be followed so that deadlock can't happen
3. Deadlock Avoidance. cond^n of deadlock arises bid escape possible.
4. Deadlock Detection.
5. Deadlock Recovery.

Deadlock Characteristics-

1. Mutual Exclusion- i. The resource has to be allocated to only one process or it is freely available
   ii. there should be a one to one relationship b/w the resource & the process.
2. Hold & Wait: The process is holding the resource & waiting on some other resource simultaneously



3. No Preemption- i. The resource has to be voluntarily released by the process after completion of the execution.
   ii. It is not allowed to forcefully preempt the resource from the process
4. Circular Wait- The processes are circularly waiting on each other for the resources



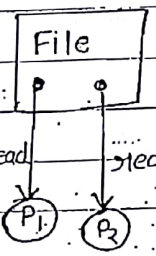
Date: / /
Page No.

- All these condition are symptoms/behavior of deadlock.

1. If all the above 4 conditions are existing in the system, than definitely there exist a deadlock.
2. All the above four conditions are not truly independent because the circular wait includes hold & wait.

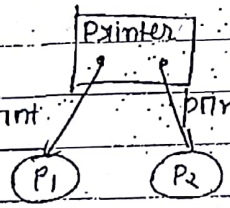
DeadLock Prevention-

1. Mutual Exclusion- It is not possible to dissatisfy the mutual exclusion always bcoz of shrable and unshrable resources



Sharable Resource

If two process want to read file at same time, they must be allowed as it is sharable resource

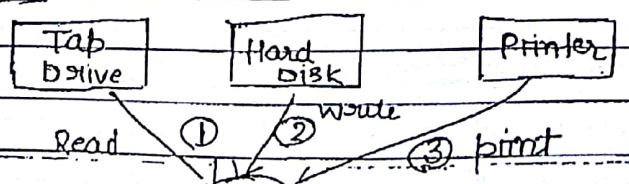


- It is not possible to allow both process to access at same time as it is non-sharable resource.

2. Hold & Wait - to avoid Hold & wait, we can do allow only holding or only waiting to avoid deadlock.

Conditions to satisfy only holding or only waiting:-

1. Allocate all the resource required by process before start of execution.



Date : / /
Page No.

• Process P_1 is reserving the resource before start of exec^n, it is basically known as low device utilization or less device resource utilization.

3. No preemption-
2. the process should release all the existing resources before making the new request.

Let P_1 request for 100 resource but only 85 are allocated to it so it cannot start its eq execution. it has to place its req- uest again for placing of new request to R_1, it has to release 85 first & then again make a request of 100 to R_1 & this time it get 70 & this will continues.

1. P_1 —req 100→ [R] , [R] —85 alloc→ P_1

2. P_1 —req 100→ [R] , [R] —70→ P_1

this may lead to
Starvation

3. No Preemption- The process P_1 is requesting for Resource R_1.

Two case arises-
1. Resource R_1 is free,
2. _____ is not free.

• If R_1 is free, it will be allocated to the process P_1.
• If Resource R_1 is not free & allocated to some other process P_2, then also two case arises-

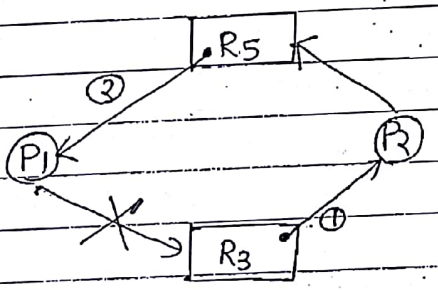
Date : / /
Page No.

(a) if the process P_2 is in execution then process P_1 has to wait.

(b) if the process P_2 is not in execution & waiting on some other process for resource R_2, than preempt the resource 'R_1' from process 'P_2' & allocate it to the process 'P_1'

4. Circular Wait - I. The resources will be assigned with the unique numerical No.
II. the process can request for the resource only in the increasing (or decreasing) order of enumeration (numbering).

If P_1 made first request for R_5 → allowed
Request II is made for R_3 by P_1 → not allowed
bcoz only increasing order is allowed.



1. P_2 request R_3 - allowed & alloted.
2. P_1 request R_5 allowed & allote
3. P_2 request R_5 → allowed
4. P_1 → request R_3 → Not allowe

Deadlock Avoidance - It will be implemented by using by using Banker's Algorithm.

~~Banker's A~~

$P_1, P_3, P_4, P_0, P_2, P_1, 3$
7, 4, 5
7, 5, 5

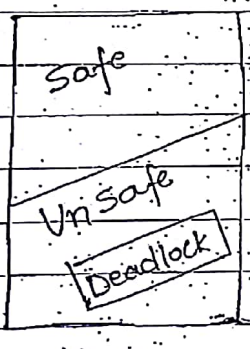
Date: / /
Page No.

Banker's Algorithm-

Total Available Resources — A ↓ 10, B ↓ 5, C ↓ 7

Process	Maximum Need A	B	C	Current Allocation A	B	C	Current Available = A	B	C	Remaining Need A	B	C
P_0	7	5	3	0	1	0	3	3	2	7	4	3
P_1	3	2	2	2	0	0	5	3	2	1	2	2
P_2	9	0	2	3	0	2	7	4	3	6	0	0
P_3	2	2	2	2	1	1	7	5	3	0	1	1
P_4	4	3	3	0	0	2	10	5	5	4	3	1
				7	2	5	10	5	7			

Remaining Need = Max Need - Current Allocation
Current Available = total - total current allocate



1. If we can satisfy the remaining need of all the processes with the current available resources than the system is said to be in safe state, otherwise, system is said to be in unsafe state.
2. if the system is in unsafe state, it is possible for deadlock.
3. the order in which we satisfy the remaining need of all the processes is called as safe sequence.
4. the safe sequence may not be unique, there may be multiple safe sequences.
5. the unsafe state purely depends on the behaviour of the process.
6. the deadlock avoidance is less restrictive than deadlock prevention.

Date: / /
Page No.

Safe Sequence → P_1, P_3, P_4, P_0, P_2
P_1, P_4, P_3, P_0, P_2
P_1, P_3, P_4, P_2, P_0
P_1, P_4, P_3, P_2, P_0

Safe Sequence (P_1, P_3, P_0, P_2, P_4)

- Whenever a process request for resources, Banker algorithm is applied to check whether deadlock is occuring or not by granting the request. if deadlock exist, request will be denied. if not, resources are allocated.

- If the system is in safe state, than the request of process will be granted & if the system is in unsafe state, than the request on the process will be denied & deadlock will be avoided.

93 Ques 2-

P_0	6	2	0
P_1	1	3	0
P_2	1	0	2
P_3	1	0	4

643.
677 / $P_3 \rightarrow$ No
644

$P_1, Q \rightarrow$ can be
$R \rightarrow$ Not

8

P_0	1	2	0	0
P_1	0	1	0	2
P_2	0	0	2	0
P_3	1	1	0	0
P_4	2	0	0	0

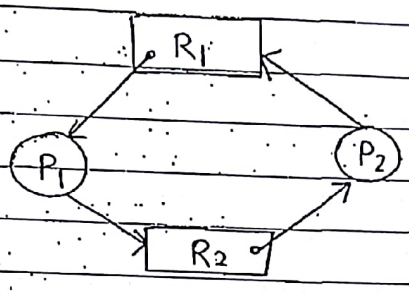
1, 1, 2, 0 — P_2, P_3, P_4, P_0, P_1
2, 2, 2, 0 exist safe state
3, 2, 3, 0 — P_2, P_3, P_4, P_0, P_1
3, 3, 3, 1
5, 3, 4, 2
6, 4, 4, 2

9. 1, 1, 2, 0
0, 0, 2, 0
1, 1, 2, 0
3, 2, 3, 0
3, 3, 3, 1
5, 3, 4, 2
6, 4, 4, 2

Date : / /
Page No.

Deadlock Detection:-

(i) the resources are of single instance type:



if all variable are of single instance type, than run cycle detection Algorithm.

$R_1 \rightarrow P_1 \rightarrow R_2 \rightarrow P_2$

- If cycle exist, ~~or no~~ then deadlock occur (necessary cycle & sufficient cond^n)
- If cycle doesnot exist, then deadlock is not there.

- If all resource are of non-single instance type, than the cycle in the resource allocation graph is just a neccessary cond^n but not sufficient cond^n for occuring of deadlock.

(ii) the resources are of multiple instance type:

	current Allocation			current Available			Remaining Need			
	A	B	C	A	B	C	A	B	C	
P_0	0	1	0	0	0	0	0	0	0	
P_1	2	0	0				2	0	2	
P_2	3	0	3				0	0	0	$\Rightarrow$ safe
P_3	2	1	1				1	0	0	
P_4	0	0	2				0	0	2	

Q What happen if the process P_2 is requesting for additional resource of (0, 0, 1)?

$P_2 \rightarrow$ old 0 0 0

Date : / /
Page No.

	Current Allocation			Current Available			Remaining Need			
	A	B	C	A	B	C	A	B	C	
P_0	0	1	0	0	0	0	0	0	0	P_0,
P_1	2	0	0	0	1	0	2	0	2	–
P_2	3	0	3	⇓ Now No other need is getting satisfied.			0	0	1	
P_3	2	1	1				1	0	0	
P_4	0	0	2				0	0	2	

$P_0 \Rightarrow$ unsafe $\Rightarrow$ Deadlock

No resources are getting allocated so P_1, P_2, P_3, P_4 remain in waiting state so slowly system is going to unsafe state. if the state of i.e resource req. of P_1, P_2, P_3, P_4 is not change than system will surely go to deadlock state. the process involved ~~will~~ in deadlock will be P_1, P_2, P_3, P_4.

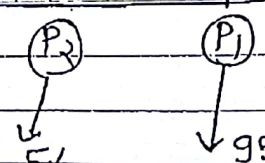
DeadLock Recovery

1. Killing the process - (a) kill all the processes which are involved in the deadlock.

(b) Kill the processes one after the other - initially a process is killed, then if still system is in deadlock than another process is kill.

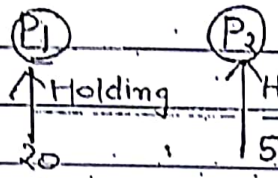
(a) low priority process - Low priority process is choosen as victim of ~~time~~ killing.

(b) % of Process completion - Process which are just started are choosen to kill



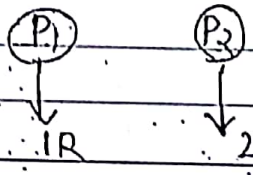
∴ P_2 get killed.

selected as victim to kill.



∴ P1 is killed.

d) No. of resource the process is requesting - the process Requesting more resource is choosen as victim to kill.



⇒ P2 is choosen.

2. Resource Preemption - i. the resources will be preempted from the processes which are involved in the deadlock and the preempted resource ~~is~~ a will be allocated to some other processes so that we can recover the system from the deadlock.

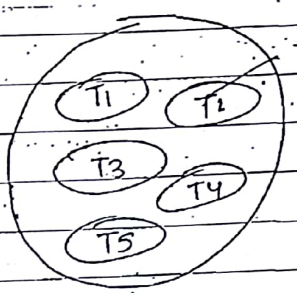
ii. if the above condition is followed than there may be a possibility for process to go into starvation.

3. Ostrich Algorithm - • ignore the deadlock.

Date : / /
Page No.

Threads - The light weight process

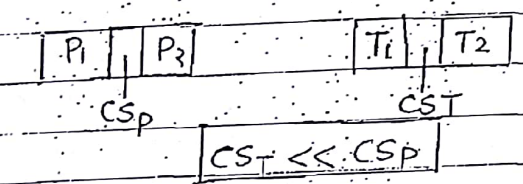
- no of instruction
- context



Advantages-

(1) Responsiveness - If the process is divided into multipl threads, than if one thread complet its execution immediately the O/P is responded this response will be fas compare to the response of the proce

2. Faster Context Switches -

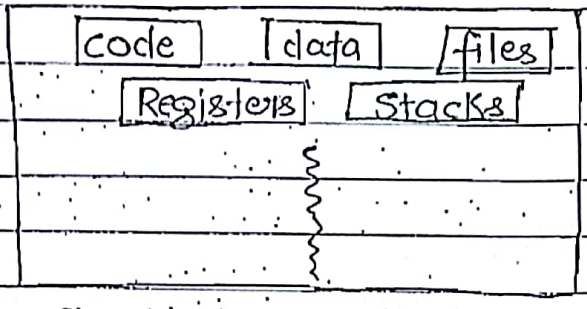


CS_T - context switchi time of threads

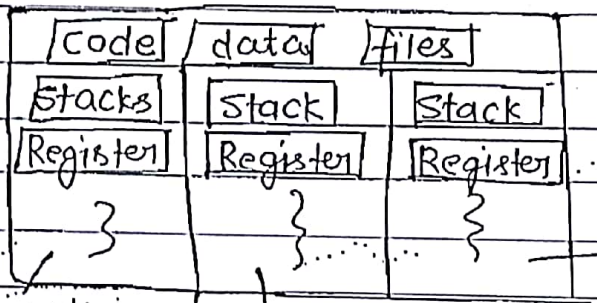
the context switching time between the threads will be very less compared to the context switching time between the processes because the threads will have less context as compare to the processes.

3. Effective Utilization of Multiprocessor system - If the proce is divided into multiple threads, than different threads can be scheduled on to different processor so that process execution will be faster.

4. Resource Sharing-



Single threaded process



Multithreaded process

- Resources like code, data, files will be shared ie memory will also be shared b/w the threads with in the process.
- But processing data structure like stacks & register will not be shared b/w the threads, as each require a separate area for processing.
- Every Thread will have its own stacks & registers.

5. Enhanced Throughput of the System- If the process is divided into multiple threads & if we consider one thread as one job then the No of jobs completed for unit time will increase & throughput of the system will be enhanced.

all threads are considered as a process to which they belong, by the OS.

Blocking system call - I/O operatic

6. Economical: Implementation of threads doesnot require any cost. there are various programming language API's which support implementation of the threa Ex - Java API.

Threads are categorized into two types -

(i) User Level Threads.
(ii) Kernal Level Threads.

User Level Threads	Kernal Level threads
1. These are implemented by user or programmer.	1. Kernal Level threads are impleme ted by the OS.
2. User Level Threads are not recognized by operating system & operating system view a thread as a process only.	2. Kernal Level threads are recognised by the operating syster
3. If one user level thread is performing blocking system call, than the entire process will go into block state.	3. If one kernal level thread is performing blocking system cal than another thread will continue the execution.
4. User Level threads are designed as dependent threads	4. Kernal Level Threads are designed as independent threa bcoz two thread can continue toge (one at I/O & other at CPU)
5. Implementation of User Level threads is easy.	5. Implementation of the kernal Level thread is complicated bc divided the instⁿ in such a way that they become independent is difficult.

Date : / /
Page No.

6. User Level thread will have less context.

6. Kernal level thread will have more context as they are independent.

7. No Hardware support required for user level threads.

7. Scheduling of kernal Level thread requires hardware support.

Note- GANG Scheduling is used in threads.

I/O operation- I/O of the process is categorized in to two types-
(i) Synchronous I/O.
(ii) Asynchronous I/O.

Synchronous I/O- 1. In synchronous I/O, the process performing I/O operation will be placed in the block state till the I/O op^n is completed.
2. Once the I/O op^n is completed, an ISR (Interrupt Service Routine) will be initiated which brings the process from Block state to Ready state.

• By defaut op^n is done using synchronous I/O.

Asynchronous I/O- 1. In this, While initiating the I/O request, the "handler function" will be registered.
2. the process is not placed in the block state & it continues to execution the remaining code after initiating the I/O request. (Remaining code not depending on I/O)
3. At the point, when I/O request is completed, the signal mechanism is used to notify the process that the data is available & registered handled function will be Asynchronously invoked.

(it is again invoked to tell about the data to the process.)

Date : / /
Page No.

Handler function - tries to specify the details of I/O operation, to be performed like which opⁿ has to be done, which data to be used.

(OS by William Stalling) MEMORY MANAGEMENT (main m/m)

- Main Memory \ Primary M/M \ Physical M/M \ RAM

Functionality of Memory Manager - Allocating & deallocating M/M to the processes.

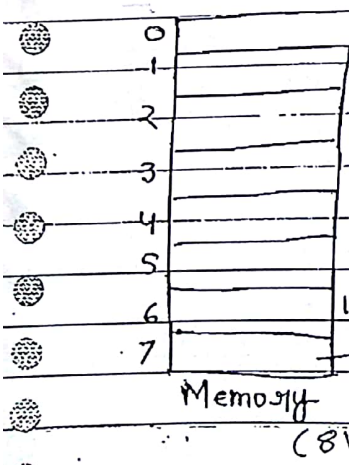
Goal: the efficient utilization of memory by minimizing the internal & external fragmentation

Memory
- Word Addressable eg 512 KW
- Byte Addressable eg: 512 KB

1 Byte = 8 bit

Basics:

$2^{10} = 1024 = 10^3 = 1K$
$2^{20} = 1M$
$2^{30} = 1G$
$2^{40} = 1T$



- Address always start with '0'.

Let word size = 16 bits

Capacity of M/M = No of word in memory × word size

= 8 × 16 bits

= 16 Bytes

Date : / /
Page No.

Ques- Consider a system which has 256 MWords & each word is having size of 128 bits what is the capacity of M/M in B?

$$\text{capacity} = 256 \times M \times 128$$
$$= 2^8 \times 2^4 \times 2^{20} \times 8$$
$$= 2^2 2^{30} \text{ Bytes}$$
$$= 4\text{GB}$$

Ques- Consider a system which has 64G words & each word is having size of 16 bytes. what is capacity of M/M in Bytes?

$$\text{Capacity} = 64 \times G \times 16 \text{ Bytes}$$
$$= 2^{10} \times G \times \text{Bytes}$$
$$= 1\text{TB}$$

Ques- consider a system where 32 Binary bits are used to represent all the words of memory & each word is having size of 32 bits what is capacity of M/M in Bytes?

$$\text{capacity} = 2^{32} \times 2^5 \text{ bits}$$
$$= 2^{34} \text{ bytes}$$
$$= 16\text{ GB}$$

RAM chip Implementation- Main M/M is implemented by using RAM chip.

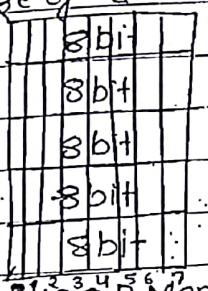
- Size of RAM chip = 128 Bytes

organize the M/M capacity of 16KB.

Date : / /
Page No.

(ii) Draw the Memory Organisation Map.
(iii) What is the size of decoder required?

RAM chip —



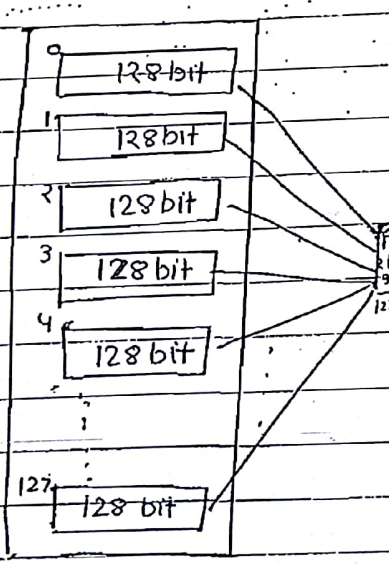
128 B Memory

(i) No of RAM chip Required = $\frac{\text{Total M/M Requirement}}{\text{Available M/M}}$

$= \frac{16\ KB}{128B}$

$= 128$

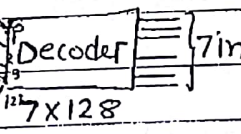
(ii)



16 kB Memory

(iii) 7 X 128

[Size of decoder = 7 X 128]



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

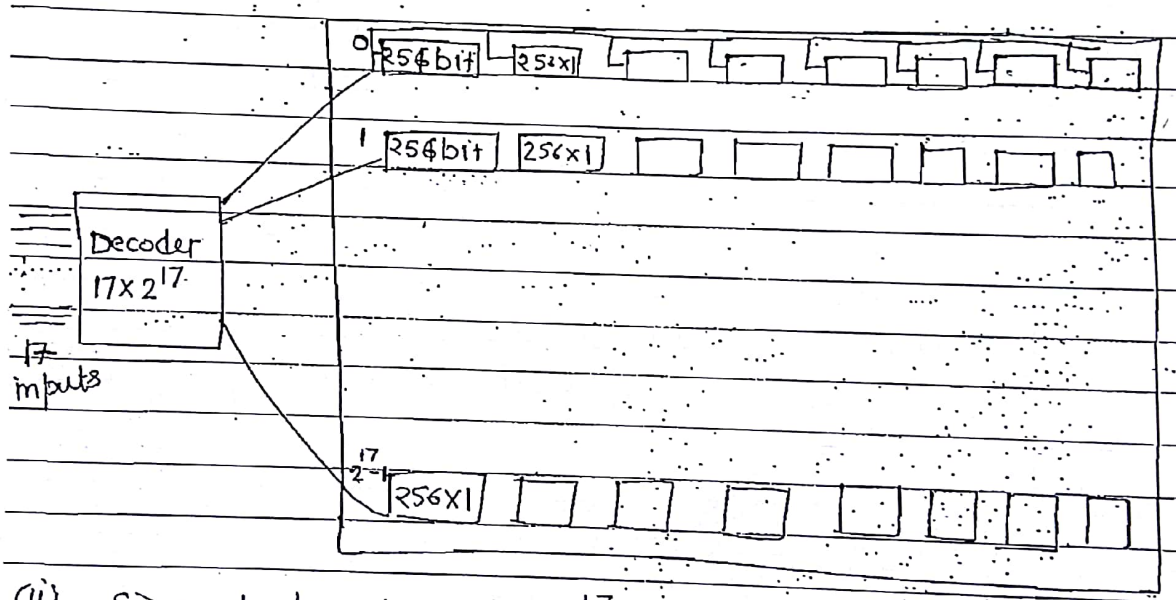
Date: / /
Page No.

Ques- RAM chip Size - 256 x 1 bit
Organize memory capacity of 32 MB.

(i) No of RAM chips Required = $\frac{32 \times 2^{20} \times 2^3}{2^8} = 2^{20}$ chips

(ii) Memory Mapping -

In a row = $\frac{2^{20}}{2^3} = 2^{17}$
In a column = 2^3



(iii) Size of decoder = 17×2^{17}

for finding row & column,

rows	32M	1 Byte column
$\frac{2^{25}}{2^8}$	256	1 bit column = 8

rows = 2^{17}

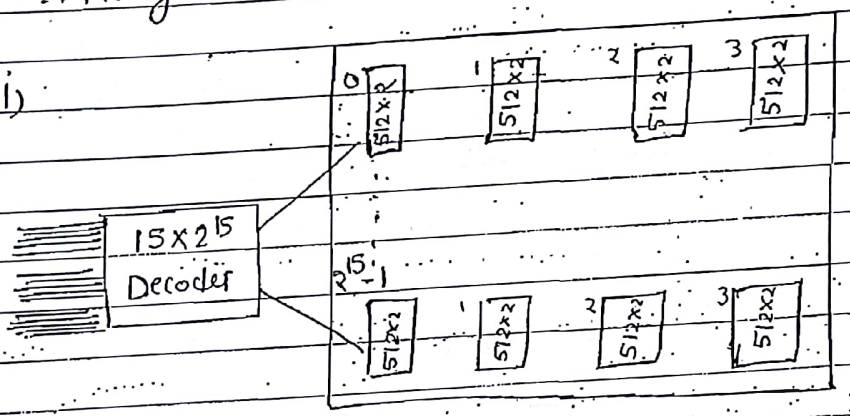
Date : / /
Page No.

Ques- RAM chip size = 512 X 2 bits
Organize the m/m capacity of 16 MB.

(i) No of RAM chip required = $\frac{16 \times 2^{20} \times 2^3}{2^9 \times 2} = 2^{17}$ chips

No of rows column → 4 → 2^2
∴ No of rows = $2^{17}/2^2 = 2^{15}$

(ii)



(iii) Size = 15×2^{15}

Ques- The main memory unit with the capacity of 4MB is build using DRAM chips. Each DRAM chip is having 1K rows of cells with 1K cells in each row. Size is DRAM chip is 1M X 1 bits, the time taken for each single refresh op is 100ns than the time required (new) to perform the refresh op^n on all the cell in M/M unit is?

Ans- No of RAM chip required = $\frac{4\,MB}{1\,Mb} = 32$ chip

for Refresh op^n, time required = 100ns

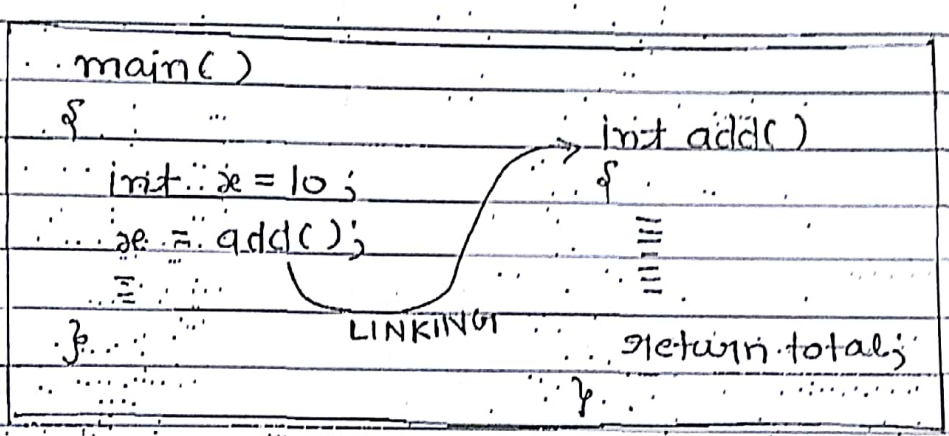
for 1 DRAM = $100 \times 1K \times 1K$
= 100×2^{20} ns

∴ for New Memory = 32 DRAM
= $32 \times 100 \times 2^{20}$ ns

(a) 100ns
(b) 100×2^{10}ns
(c) 100×2^{20}ns
(d) 3200×2^{20}

Loading- Bringing the program from secondary memory to main memory is called as Loading.

Linking- Establishing the linking between all the modules or all the functions of the program in order to continue the execution.



Loading & Linking is further categorized into two types-

1. Static.
2. Dynamic

Static :1. Loading the entire program into memory before start of program execution is known as static Loading.

2. Inefficient utilization of Memory because whether it is required or not required, the entire program will be brought in to memory.

3. The program execution will be faster.

4. If the static loading is used accordingly, the static linking will be applied.

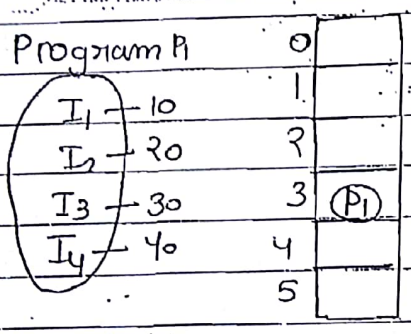
Date : / /
Page No.

Dynamic :- 1. Loading the program into memory on demand is called as dynamic loading.
- Initially only main program function is loaded to main memory, when add f^n is needed while execution than it is bought to the main memory i.e f^n are getting added to main m/m on demand.

2. The efficient utilization of memory is carried out.
3. The program execution will be slower.
4. if the dynamic loading is used, accordingly the dynamic lnkg will be applied.

Address Binding - Association of program inst^n & data to the actual physical M/M Location is called as Address Binding.

2. It is perform only after linking & loading of program.



each instruction of a program is getting binded with an address so that its access become easy.

PC = 10 20 30 40

Three types - 1. compile time Address Binding } theoritical concept.
2. Load time
3. Execution time AB or Dynamic AB.

Compile time Address Binding - 1. If the compiler is responsible of association of program instruction & data to the actual physical M/M location is called as compile time Address Binding

3. The compiler requires to interact with a OS memory manager to perform compile time address binding.

2. Load Time Address Binding - This type of address binding will be done after loading the program into memory.

2. This type of address binding will be done by OS memory manager (Loader).

3. Execution / Dynamic Address Binding - 1. the address binding will be postponded even after loading the program in to memory.

2. The program will keep on changing the locations in the memory till the time of program execution.

3. This type of Address Binding will be done at the time of program execution & this will be done by the processor

- practically used approach.

- postponing of address binding is done bcoz it keep on changing its position due to memory mgt technique by os.

- ~~Re~~ ~~Practi~~ Note - Majority of the OS practically implements dynamic loading, dynamic linking & dynamic address Binding.

Date : / /
Page No.

Memory Management Techniques-

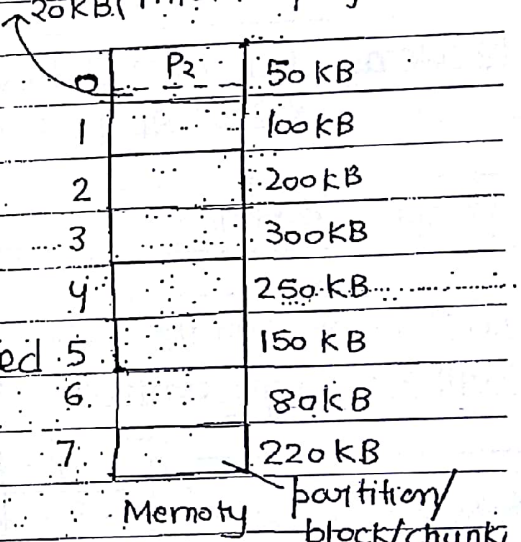
(i) Contiguous technique
- fixed partition scheme
- Variable partition scheme

(ii) Non contiguous technique
- Paging
- Multi Level Paging
- Inverted paging
- Segmentation
- Segmented Paging

Contigous M/M Mgt Technique-

(a) Fixed Partition Scheme-

- No. of partition is fixed.
- Size may vary: (size of partition)

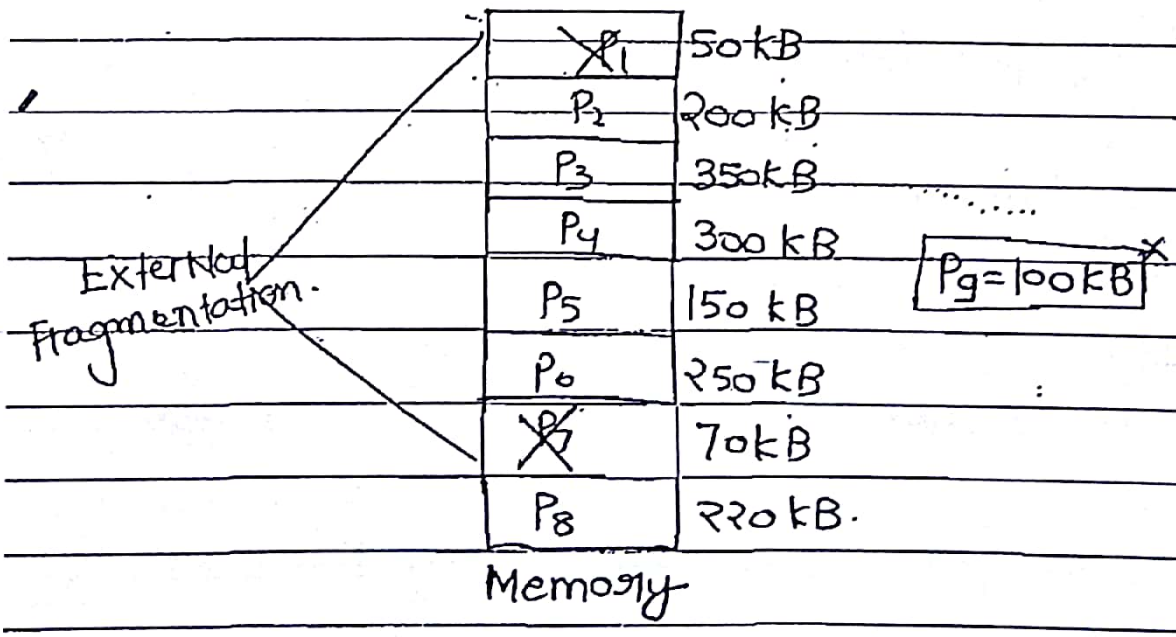


1. In this scheme, the M/M will be divided in to fixed No of partitions.
2. Fixed means the No of partitions are fixed in the memory.
3. In this, in every partition, only one process will be accomodated.
4. the degree of multiprogramming is restricted by No of partitions in the main memory. ie only N process are entertained.
5. Maximum size of the process is restricted by maximum size of the partition.
   if $P_1 \rightarrow 350$ kB this cannot be accomodated.
6. if $P_2 = 30$ kB reach M/M it is placed in 50kB partition & 20kB is getting wasted, known as internal fragmentation (it lead to fragmentation as this 20kB cant be used to enter

Problem- 1. Degree of Multiprogramming is restricted.
2. Size of process is restricted with size of p

Variable Partition Scheme-

- In Variable Partition Scheme, initially memory will be single continous free block.



- Whenever the request by the process arrives accordi ition will be made in the memory.
- Initially $P_1 \Rightarrow$ 50kB, than a 50 kB partition in m & P_1 will be accomodated there.

Date : / /
Page No.

- Here problem of restricted degree of multiprogramming is resolve by not fixing the No of partition.
- problem of accomodating larger size process also get resolved
- the problem of internal fragmentation is also reduced.

Now, $P_2 \Rightarrow 200\,kB$
$P_3 \Rightarrow 350\,kB$
$P_4 \Rightarrow 300\,kB$
$P_5 \Rightarrow 150\,kB$
$P_6 \Rightarrow 250\,kB$
$P_7 \Rightarrow 70\,kB$
$P_8 = 220\,kB$

Suppose after some point of time, suppose P_1 completed, then OS will perform task of deallocation of Memory:
Suppose P_7 also get completed than it also get deallocated

After DeAllocation,

free space = 120 kB

Let P_9 arises $P_9 \Rightarrow 100\,kB$

this request cannot be entertained as process is to stored in continous form.

This problem is known as External Fragmentation.

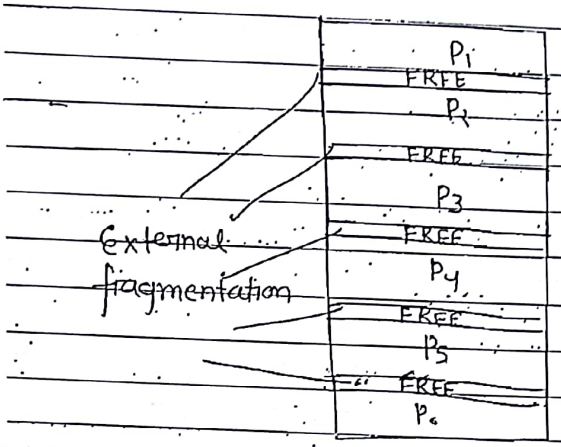
External Fragementation - problem of not being able to use available memory as it is not contina

- Let $P_{10} = 30\,kB$ ∴ it is placed to 50 kB blocked then we get a partition of 20 kB.

If in this manner, small process remain coming than small free slot are available. these small slot lead to external fragmentation for large processes.

Date : / /
Page No.

• If the smaller processes keep on coming than larger partition will be made into smaller partitions.



these small partition can not accomodate a large process as contigious entity. Hence leads to external fragmentation.

To avoid the external fragmentation, the following technique are used:

(i) Compaction: 1. Moving all the process towards top or towards the bottom to make the free available memory in a single continous place is called as compaction.
2. the problem is that all the process get disturbed.
3. It is not suggestible to implement it practically because it interrupts all the running processes in the memory.

(ii) Implementing Non contiguous memory mgt technique.

(a) Partition Allocation Methods= it basically select the ~~(a) Best Fi~~ partition for a process to be placed when more than one space is available.

(i) First Fit: Allocate the process in a partition which is first sufficient partition from top of the memory.

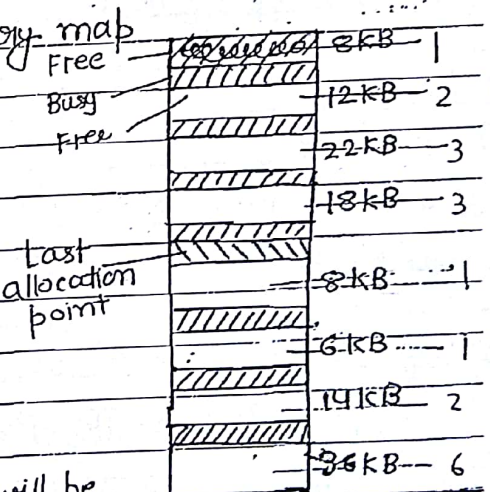
Date: / /
Page No.

2. Best Fit: 1. Allocate the process in a partition which is the smallest sufficient among the free available partitions.
2. To find out the smallest sufficient, it requires to search all the free partitions in the memory.

3. Worst Fit: 1. Allocate the process in a partition which is largest sufficient among the free available partitions.
2. To find out the largest sufficient, it requires to search all the free partitions in the memory.

4. Next Fit: The Next Fit also works like the first fit but it will search for the first sufficient partition from the last allocation point.

Ques - Consider the following memory map



(i) First Fit: P_1 = 22 KB partition
(ii) Best Fit: 18 KB for P_1
iii) Worst Fit: 36 KB for P_1
iv) Next Fit: 36 KB for P_1

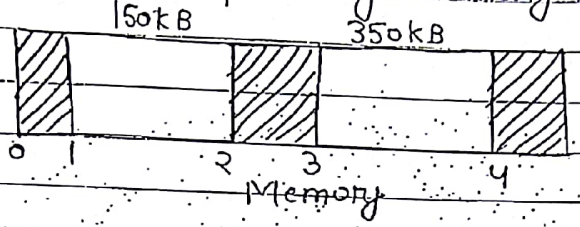
$P_1 \Rightarrow 16$ KB

How many successive request of 6 KB will be satisfied By using first fit assumming variable partition scheme?

Memory 19

Date : / /
Page No.

Ques- Consider the following memory map



The request from the process are 300 kB, 25 kB, 125 kB, 50 kB respectively. the above request can be satisfied with (Assume Variable partition technique)

(a) Best fit but NOT first fit

(b) first fit but NOT best fit

(c) Both first fit & best fit

(d) Neither first ~~first~~ fit NOR best fit

the Best fit not always result the best solution for the problem.

Non Contiguous Memory Mgt. Technique- · process need not be allocated a continous memory.

1. Paging-

TERMINOLOGY-

1) Logical Address Space or Virtual Address Space (LAS or VAS)-
- Represented in the form of words or bytes Eg- 32 MW or 32 MB

2. Logical Address (LA) or Virtual Address (V.A)-
- Represented in the form of bits Eg: 25 bits or 32 bits

3. Physical Address Space (P.A.S)-
- Represented in the form of word or bytes Ex- 512 KW or 512 KB

4. Physical Address (P.A)-
- Represented in the form of bits Ex- 19 bits

Date : / /
Page No.

Ques if Logical Address is of 38 bit what is Logical Address Space

Logical Address Space $= 2^{38}$ word
$= 256$ G Words

Ques- if Logical Address space is 64 T Word what is logical address?

Logical address space $= 64$ T word
$= 2^6 2^{40}$ word
$= 2^{46}$ word

∴ **Logical Address = 46**

Ques P.A. = 37 bits
what is P.A.S = ?
P.A.S $= 2^{37}$ Word
$= 128$ G word

Ques- if P.A.S = 32 KB B - for byte addressable memory
P.A. = 15 bits no need to include it.

NoTE- CPU always generate 'Logical Address' in OS.

Paging - 1. The technique of mapping CPU generated logical address to physical Address is called as paging.
2. the paging will be implemented at hardware level.
3. If the logical address is same as physical address, than No hardware support of paging is required.

Basically when a process execute, it is loaded to RAM & get PA, but CPU generate logical address for ex^n of instruction. both PA & LA is not same, generally

Date : / /
Page No.

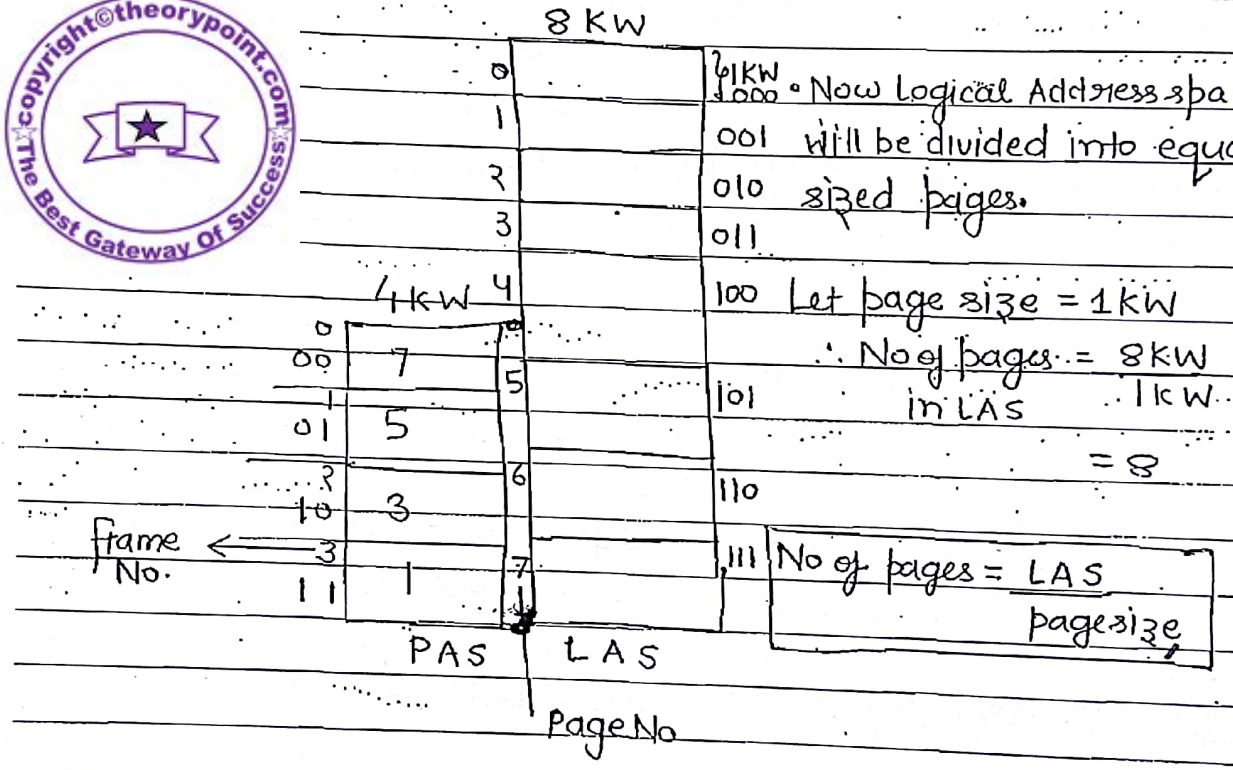
Paging:- Let L.A = 13 bits
P.A = 12 bits

Here **LA > PA**

L.A.S = 2^{13} W
= 8 kwords

P.A.S = 2^{12} W
= 4 kWords

Let system is word Addressable



- Now Logical Address space will be divided into equal sized pages.
- Let page size = 1 kW
- No of pages = 8 kW / 1 kW in LAS = 8

No of pages = LAS / page size

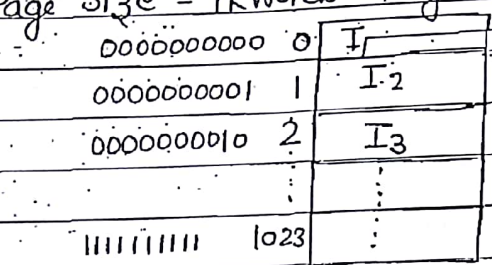
- Physical Address space will be divided into ~~blocks~~ frames of equal size
- Page size will always same as frame size

{Page Size = Frame Size}

No of frames = PAS / frame size = 4 kW / 1 kW = 4

Date : / /
Page No.

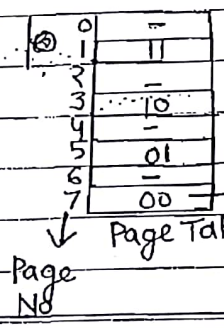
- Some of the Pages of LAS will be brought into PAS. ~~this is~~
- Let Page No 7, Pg No 5, P3, P1 are brought to PAS.
- Page Size = 1k words. Page will have 1 k W



→ stores one word in it
(Here actual instruction is getting stored)

1kW → require 10 bit for addressing

- Whenever the paging is applied, the page table will be maintaine

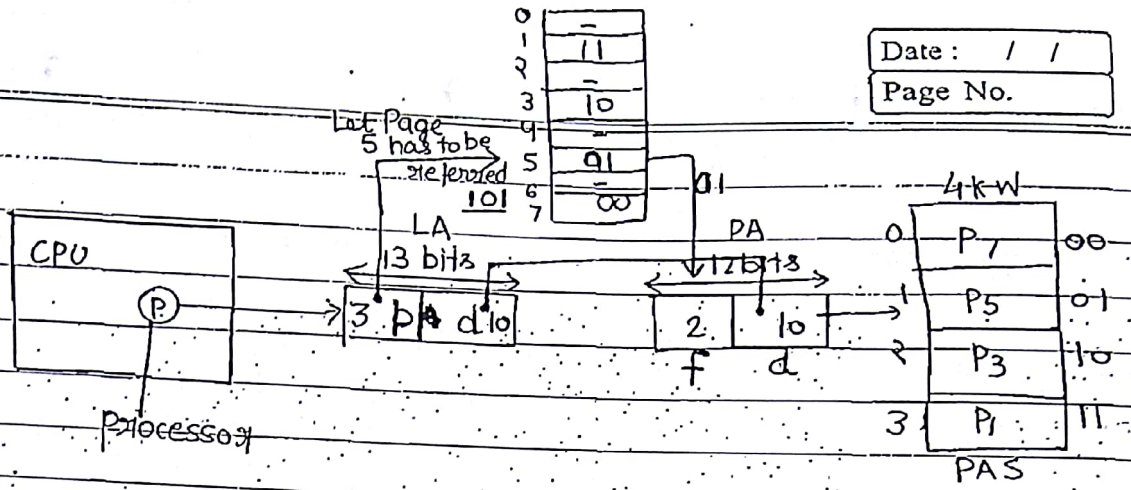


- it include the entry equa to No of pages.

→ Page Table Entry (P.T.E)

- No of pages in the LAS is equal to the No of entries in the Page table.
- Page entry ~~is~~ contains frame No definitely & can store some add bit information.
  as P1 is ~~also~~ available in ~~Page~~ Frame 3
- the page table entry definitely contains frame No.
- Page Table is also called as Address Translation table.
- the frame No is also called as Translation bits.

Date : / /
Page No.



p = No of bits required to represent pages of LAS or page No.

d = the No of bits required to represent page size or the word no of the page.

- It is also known as page offset (d)

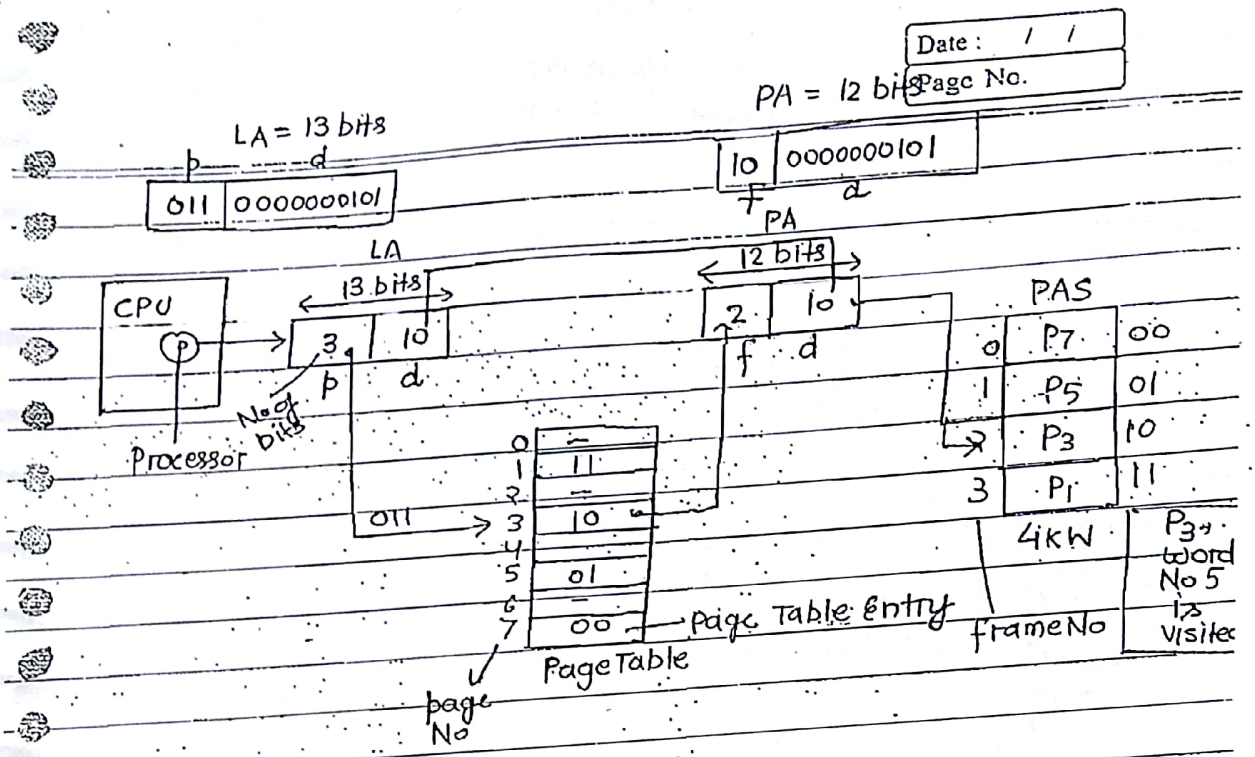
- we assume that physical Address is 12 bits.

f = No of bits required to represent frames of PAS or frame No

d - the No of bits required to represent frame size or the word No of the frame or frame offset.

- p is used access the page table & corresponding frame No is obtained.
- d is directly copied to the P.A. Now the f data will tell about the frame No to be visited & d will help in selecting word from page.

Date : / /
Page No.



Important Points-

1. Whenever the process is created, the paging will be applied on the process & page table will be created, and the base address of the page table will be stored in the PCB (process control Block).

2. Paging is with respect to every process & every process will have its own page table.

3. the page tables of the processes will be stored in the main memory.

4. There is no external fragmentation in the paging bcoz frame size is equal to page size as each page get fitted into frame.

5. The internal fragmentation exist in the last page & internal fragmentation in the paging is considered as $\frac{P}{2}$ where 'P' is the page size.

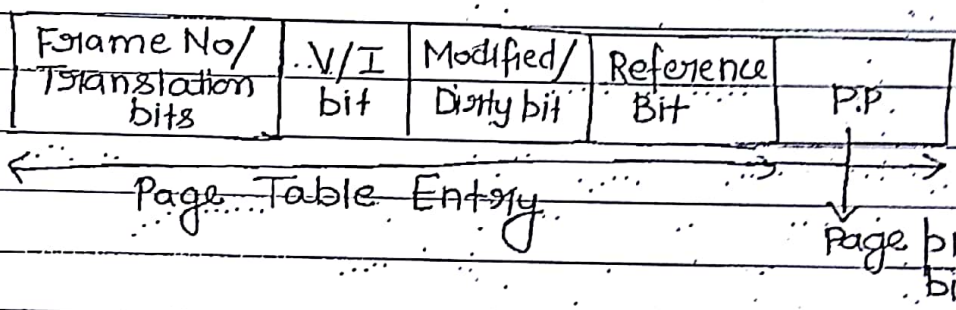
(P) half

Ex- LAS = 16kW ∴ No of page = $\frac{16kW}{3kW}$ = 5.33 ≈ 6

page size = 3kW

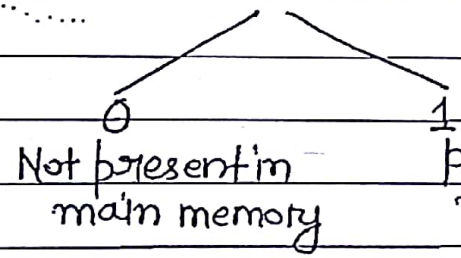
NOTE- the page table entry definitely contains frame No but sometimes, along with the frame No., it may also contain add^n bits. like-

(i) Valid / Invalid bit or Present Bit.
(ii) Modified Bit or Dirty Bit.
(iii) Reference Bit.
(iv) Page Protection Bits

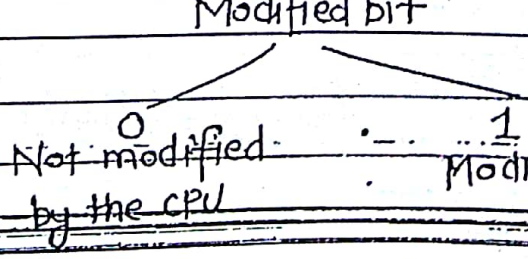


Valid/Invalid bit - it is used to identify whether the page is currently available in the main memory or not.

It takes two values -



Modified bit or Dirty Bit - it is used to identify whether the page is modified or not modified while accessing the page by the processor as part of the execution



Date : / /
Page No.

Reference bit- the Reference bit is used to identify how many times the page is referred by while performing the execution.

- Any No of bit is used show it.

Page Protection Bits- these bits are used to provide protection & security from the unauthorized access

Ques- Consider a system which has logical address = 27 bits & physical address is 21 bits & page size is 4KW. than, calculate No of pages & No of frames

$$LAS = 2^{27}W = 128GW$$

$$PAS = 2^{21}W = 2GW$$

$$\text{No of pages} = \frac{128 \times 2^{20}}{2^2 \times 2^{10}} = 2^5 \times 2^{10} = 2^{15} \text{ pages}$$

$$\text{No of frames} = \frac{2 \times 2^{20}}{2^2 \times 2^{10}} = 2^9 \text{ frames}$$

Ques- Consider a system which has no of pages = 8K & page size is 16kB. physical address is 18 bits & memory is Byte Addressable than calculate Logical address & No of frames.

$$LAS = 16 \times 2^{10} \times 2^{10} \times 2^3 \Rightarrow 2^{27}B$$

$$LA = 27$$

$$\text{frame} = \frac{2^{18}B}{16kB} = 2^4 = 16$$

∴ Logical Address = 27
frames = 16

Date : / /
Page No.

Consider a system which has LAS = 128 MW & PA = 24 bits the PAS is divided into 8 k frames the memory is word addressable than what is page size & LAS & how many pages are there in LAS

$$8 \cdot 8K = \frac{2^{24} W}{x W}$$

$$x = \frac{2^{24}}{2^{13}}$$

$$x = 2^{11} W = 2 KW$$ page size = 2kW

No of pages = $\frac{128 MW}{2 K.W} = 64 K$ pages

Ques Consider a system which has LA = 32 bits & PAS = 64 MB & page size is 4KB. than what is the approximate size of page table in Bytes

$$\text{No of pages} = \frac{2^{32} B}{2^{12} B} = 2^{20} \cdot 2^{20}$$

$$\text{No of frame} = \frac{2^{26}}{2^{12}} = 2^{14} \quad \underline{14}$$

$\underline{2MB}$

No of bits required in page table entry = 14 ≈ 16

$$\text{total size} = 2B \times 2^{20}$$

$$= 2MB$$

{ P.T size = # of Entries in P.T × P.T.E size }

Date : / /
Page No.

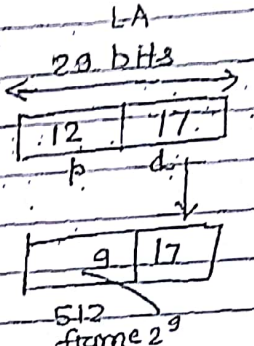
Ques- Consider a system having the page table with 4K entries and logical address is 29 bits than what is the physical address if the system has 512 frames?

$LAS = 2^{29} W$

$4K = \frac{2^{29}}{x}$

$x = \frac{2^{29}}{2^{12}} = 2^{17}$

page size = 128 KW



∴ 26 bits

$512 = \frac{PAS}{128K}$

$PAS = 2^{26} W$

PAS = 26 bits

3g- $LAS = PAS = 2^{16} B$

Page size = 512 B

No of frame = $\frac{2^{16}}{2^9} = 2^7$

16 = 1+1+1+3+7

7 bits for frame

1+1+1+3 → for extra

∴ Remaining = 16 - 7+6 = 16-13 = 3

Ans-

Ques- Consider a system which has logical address of 40 bits & page size is 16 KB & page table entry require 48 bits than what is the page table size ____ in Mega Bytes

No of pages = $\frac{2^{40} B}{2^{14} B} = 2^{26}$ pages

∴ page table = 6×2^{26} Bytes

$= 2^6 \times 6$ MB

$= 64 \times 6$

$= 384$ MB

Date : / /
Page No.

GATE 2015

Consider a system which has physical address of 32 bits & page size of 8KB & memory is byte addressable. the page table entry contains a valid Invalid bit, a dirty bit, & 3 permission bits & the translation bits. the page table size of a process is 24 MB then what is length of logical address.

$$\text{No of frame} = \frac{2^{32}B}{2^{13}} = 2^{19}$$

frame = 19 bits

1 page table entry = 24 bits

No of pages = $1MB = 2^{20} \times 2^3$

∴ Logical Address space = $2^{20} \times 2^{10} \times 2^3 B \times 2^3$

$= 2^{38+3}B = 2^{36}$

Logical Address = $2^{36} = 36$

16-oct-2016 -

Ques- Consider a system which has LAS = PAS = 's' bytes & page size = p bytes & page table entry size is e bytes. the memory is byte addressable. what is the optimal value of page size by minimizing the memory overhead of maintaining the page table size & internal fragmentation in paging?

page table size = $\frac{s \times e}{p}$ $\frac{s}{p}$ must be less

for internal fragmentation

Date : / /
Page No.

Here page table size = $s/p \times e$

$P/2$ internal fragmentation in $s/p \times e$

for minimum overhead, $\frac{P}{2} = \frac{s \times e}{P}$ (suspect)

$P^2 = 2se$

$P = \sqrt{2se}$

Memory Overhead = Page table size + Internal fragmentation in paging

$= \frac{s}{p} \times e + \frac{P}{2}$

for minimum overhead = 0 $\frac{s}{p} \times e + \frac{P}{2}$ $\frac{dM}{dP} = 0$

$\frac{1}{2} - \frac{se}{P^2} = 0$ $P^2 = 2se$

$P = \sqrt{2se}$

$\therefore \boxed{P = \sqrt{2se}}$ Ans.

Performance of Paging - 1. The main memory access time = M

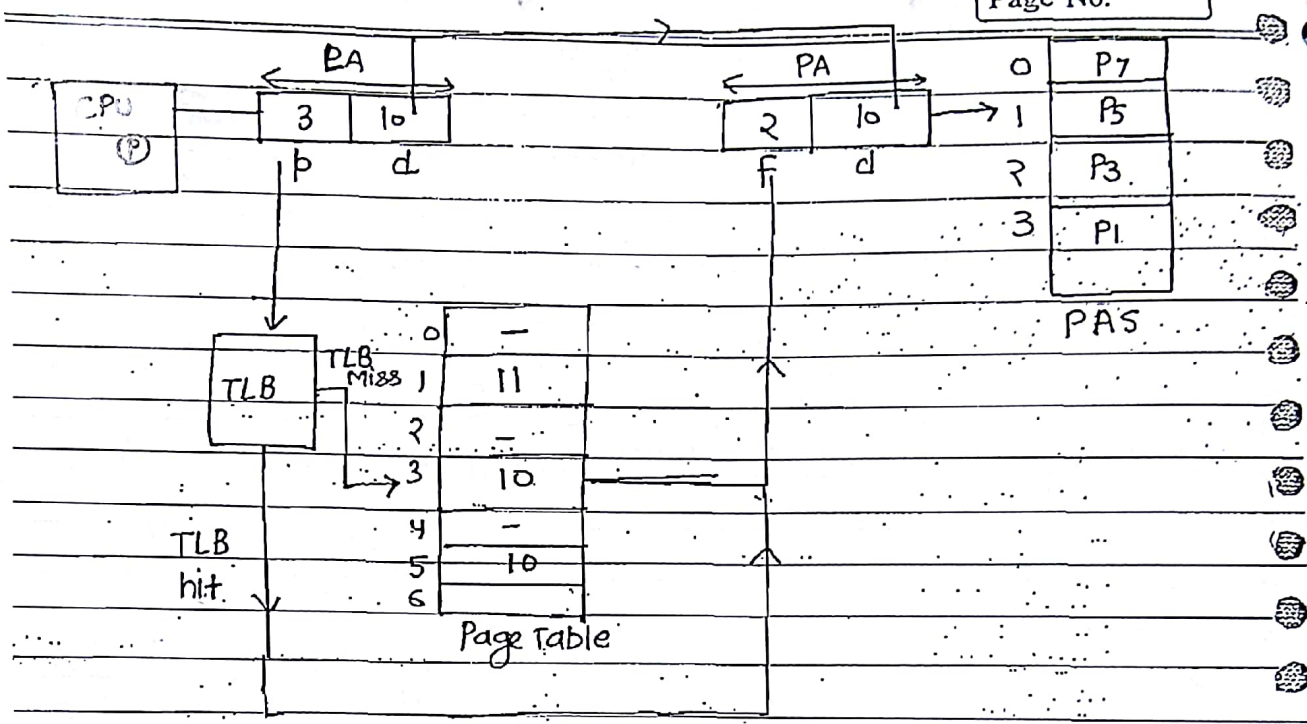
2. the page tables are stored in main memory

than, the formula for effective M/M Access time,

$$\boxed{E.MA.T. = 2M.}$$

1. The translation look Aside Buffer (TLB) is added to improve th performance of paging.
2. The TLB is a hardware device which is implemented by associative Registers.
4. The TLB contains frequently refferred page No & correspondi frame No.
5. The TLB access time will be very less compared to the main M/M

Date : / /
Page No.



- Initially system will Look in TLB for page. If page No is there than corresponding frame No is taken, known as TLB Hit & directly go to PA.
- If page is Not there, Now it again look for page in page table residing in main m/m.

- The TLB Access time = C
  TLB Hit Ratio = α
  the formula for effective M/M access time,

$$EMAT = \alpha(M+c) + (1-\alpha)(2M+c)$$

Ques Consider a system which has main m/m access time = 100ns & TLB Access time is 20ns & TLB hit ratio is 95% than what is effective M/M access time with TLB & without TLB?

Date : / /
Page No.

With TLB, EMAT = 0.95 (120) + 0.05 x 220
= 114 + 11
= 125 ns

Ques- How much hit ratio is required to reduce the effective memory access time from 300ns w/o TLB to 250 ns with TLB? The TLB access time is 60 ns.

Ans- $300 = 2 \times M$

$M = 150$ ns

$250 = \alpha(150 + 60) + (1-\alpha)(2 \times 150 + 60)$

$25 = 21\alpha + 36 - 36\alpha$

$15\alpha = 36 - 25$

$\alpha = 11/15 = 0.73.3$

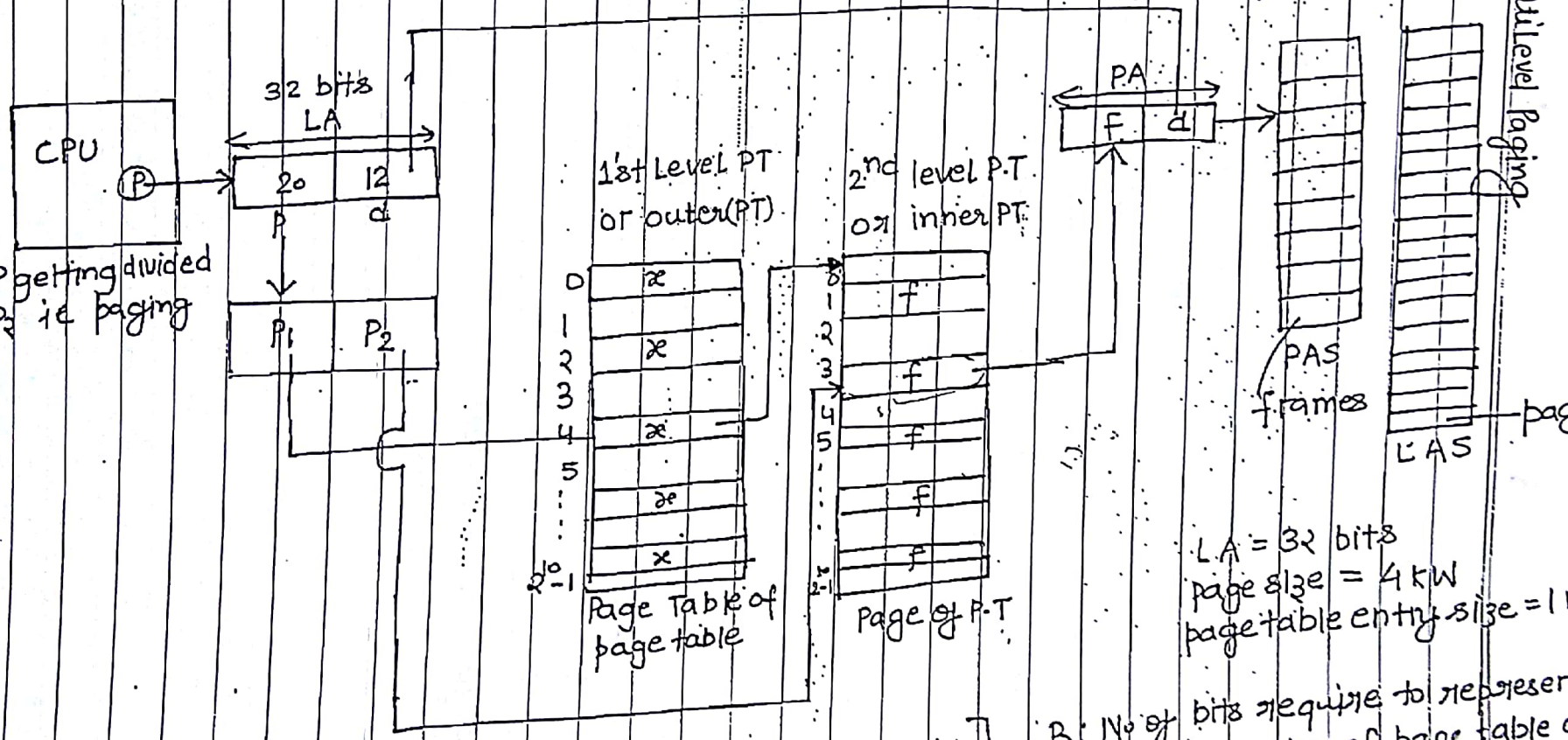
73.3 %.

Ques- Consider a system which has logical address = 32 bits & page size is 4 KW & page table entry size is 1 word & Memory is word addressable than what is page table size?

No of page $= \frac{2^{32} W}{2^{12} W} = 2^{20}$

∴ Page table Size $= 2^{20} \times 1$ word
= 1 M word

Multilevel Paging

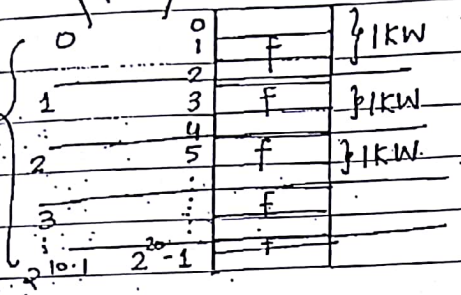


Page No.

Date: / /

Date : / /
Page No.

Here, Page Table Size = $\left(\frac{2^{32}}{2^{12}}\right) \times 1W = 2^{20}$ word = 1MW



1. To avoid the overhead of bringing large size page table into memory, the multilevel paging will be implemented.
2. In the Multilevel paging, paging will be applied on the page table.
3. Instead of Bringing the entire page table into memory, the pages of page table will be brought into memory.

ie Now page table is divided into pages of size 1KW

No of pages in page table = $\frac{1MW}{1KW}$ = size of page table / size of page

= 1K

Page table of page table - Representation of pages of page table inside main memory.

- P_1 is used to know which page of page table has to be referred. $P_1 = 4$ than 4th page is reffered which store x. than we will go to address x & table at x is loaded to main memory & by using P_2, we get the actual frame No..

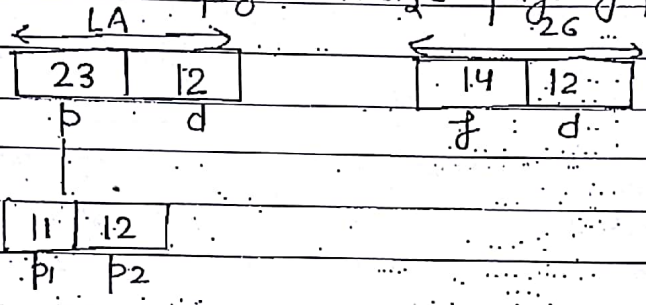
Here overhead of memory actually decreases bcoz if Multi-Level paging is not done, than whole table of 2^{20} size is to be loaded to main m/m but now it is efficient as by using it only two tables of 2^{10} size are getting loaded.

Date : / /
Page No.

Ques- Consider a system using 2 level paging applicable. the page table is divided into 2K pages and each page is having 4K entries. the PAS is 64 MW which is divided into 16K frames the Memory is word addressable & page table entry size in both ~~wo~~ the level is 2 word. calculate.

(i) length of Logical Address.
(ii) length of physical ____
(iii) First Level page table size
(iv) Second Level page table size - (page of page table)

Ans -

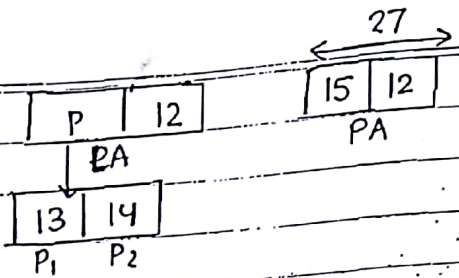


(i) Length of logical address = 35
(ii) Length of physical address = 26
(iii) First Level page table size = $2^{11} \times 2$
$= 2^{12} W$
$= 4 KW$
(iv) Second level size - $2^{12} \times 2$
$= 8 KW$

Ques Consider a system which has 2-level paging level applicable. the page table is divided into 8-K pages & each page is having 16K entries the PAS is 128MB which is divided into 4~~8~~KB frames. the m/m is byte addressable & page table entry size in both level is 32 bits. Calculate

(i) Length of LA

Date : / /
Page No.



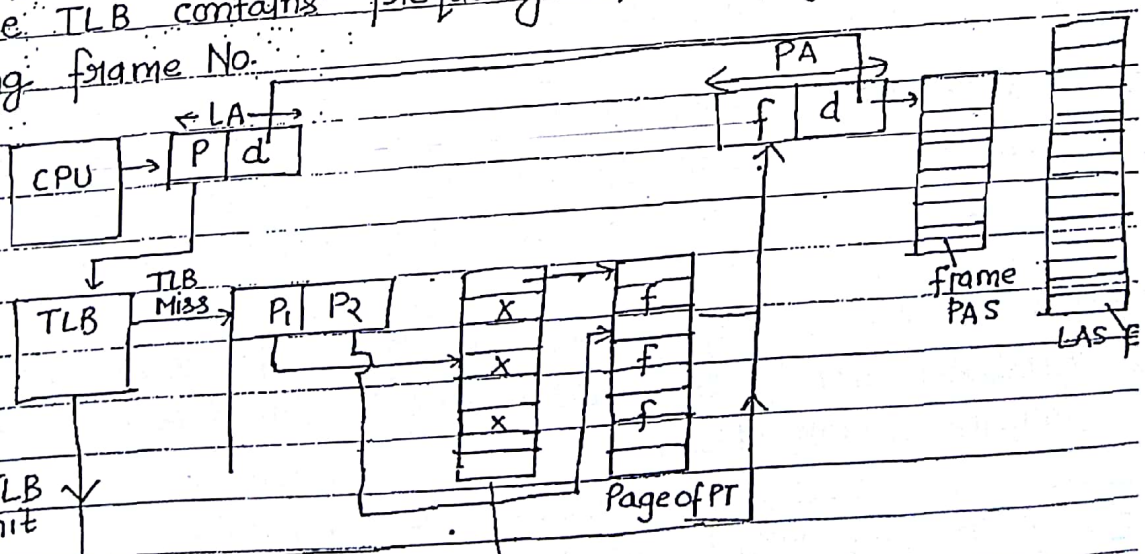
(i) Length of LA = 39
(ii) Length of PA = 27
(iii) PT first level = $2^{13} \times 32$ bits = 2^{15} Bytes = 32 kB
(iv) PT second level = $2^{14} \times 32$ bit = 64 kB

Performance of two level paging -

The main memory access time is m.
the page tables are stored in the main memory than, the formula for effective memory access time,

$$EMAT = 3m$$

1. The Translation LookAside Buffer TLB is added to improve the performance of paging.
2. The TLB contains frequently referred page No's & corresponding frame No.



Date : / /
Page No.

The TLB Access time = c
TLB hit ratio = α
than the formula for Effective Memory Access time, for 2 Level paging

$$E.M.A.T = \alpha(M+c) + (1-\alpha)(3M+c)$$

In this manner
for n-level paging,

$$E.M.A.T = \alpha(M+c) + (1-\alpha)(c+(n+1)M)$$

Important Points- 1. In the Multilevel paging, when the paging applied on the page tables, the last page table which we get is called as first level page table.

2. In the Multilevel paging, when the paging applied on page table, the first level page table entry contain base address of second level page table, the second ~~table~~ level page table entry contain base address of 3 level page table and so on & the final page table entry contains frame No. of actual page. ~~In a M~~

3. In a Multilevel paging, when the paging applied on page tables whatever may be the levels of paging all the page tables (page of page table) will be stored in main memory.

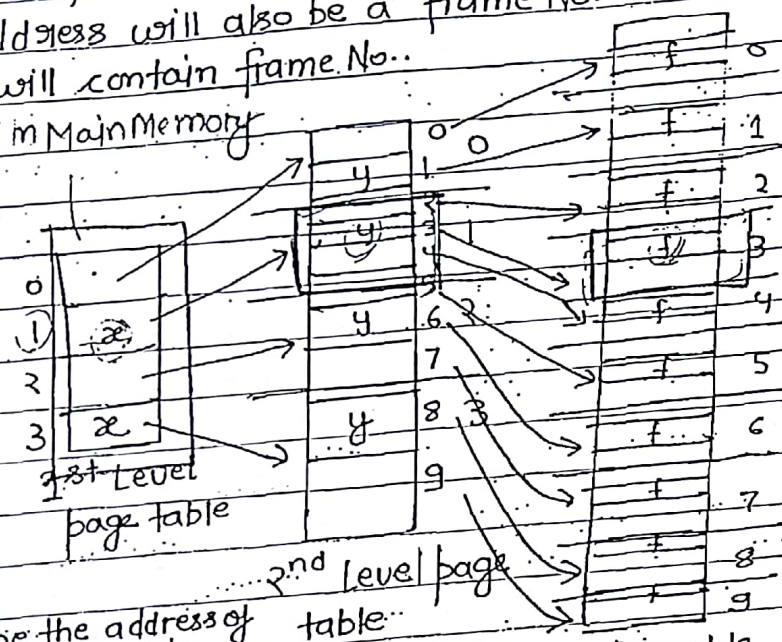
4. In a multilevel paging, when the paging applied on page tables, whatever may be the levels of paging, the page table entries contains frame No.

5. If the page size is not mentioned in the problem, Generally page size will be same in all the places (Levels)

[Windows, page size = 4KB]

Date: / /
Page No.

4. point- page tables entry will contain base address of page of page table. but that page of page table is also stored in main m/m i.e stored in a particular frame i.e that base address will also be a frame No. Hence, all the page table will contain frame No.. will stored in Main Memory



- x - will be the address of page 1 in 2nd level page table
- y - will be address of page 2 in 3rd level page table

- Let us suppose for page 1 in 1st Level, ∴ 1 level table & 1 page of 2nd Level table & 3rd page of 3rd level table get stored in Main M/M.

- No of bits in x = No of bits in y = No of bits in f

as all the three thing are present in MM. since main memory is represented with frames

∴ x, y, f will store some frame No.

Main Memory All the page table are stored in M/M

P_1	P_2	P_3	d

→ 3rd level PT index

1st Level PT indexing

2nd level PT indexing

Ques 25 Page Size = 1KB

LA = 32

PTE = 4B

No of pages = $\frac{2^{32}}{2^{10}} = 2^{22}$

22	10

↓

10	10	10

$1 + \log 22$

$5 \log 5$ 3 4

2^{24} 2^{10}

2^{24} pages 256

PT = 2^{26}

PT size = 2^{29}

Each page table must fit with in a page ie its size must be equal to page size.

ie 1024 bytes

for 1 table- for one entry - 4 byte

∴ No of pages allowed - 256

∴ No of bits = 8

∴ 22

8	8	6

P_3 P_2 P_1

∴ No of Levels = 3

(first level page table size)

Sir

1st time paging- Page table size = $\frac{2^{32}}{2^{10}} \times 4B$

Date : / /
Page No.

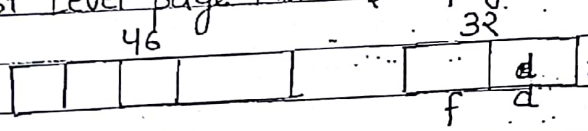
Ques - GATE 2013 - Consider a system which has logical address of 46 bits and physical address of 32 bits and memory is byte addressable & page table entry size is 32 bits the operating system uses 3-level paging for logical to physical address translation and first level page table size is exactly same as page size than what is the page size?

LA = 46 bits

PA = 32 bits

PTE = 32 bits = 4B

First Level page table size = page size = x



$x = PTE_1 \times 4$

$PTE_1 = x/4$

32 bits for frame

2^{32} 2^{32}

I paging - page table size $= \left(\frac{2^{46}}{x}\right) \times 4$

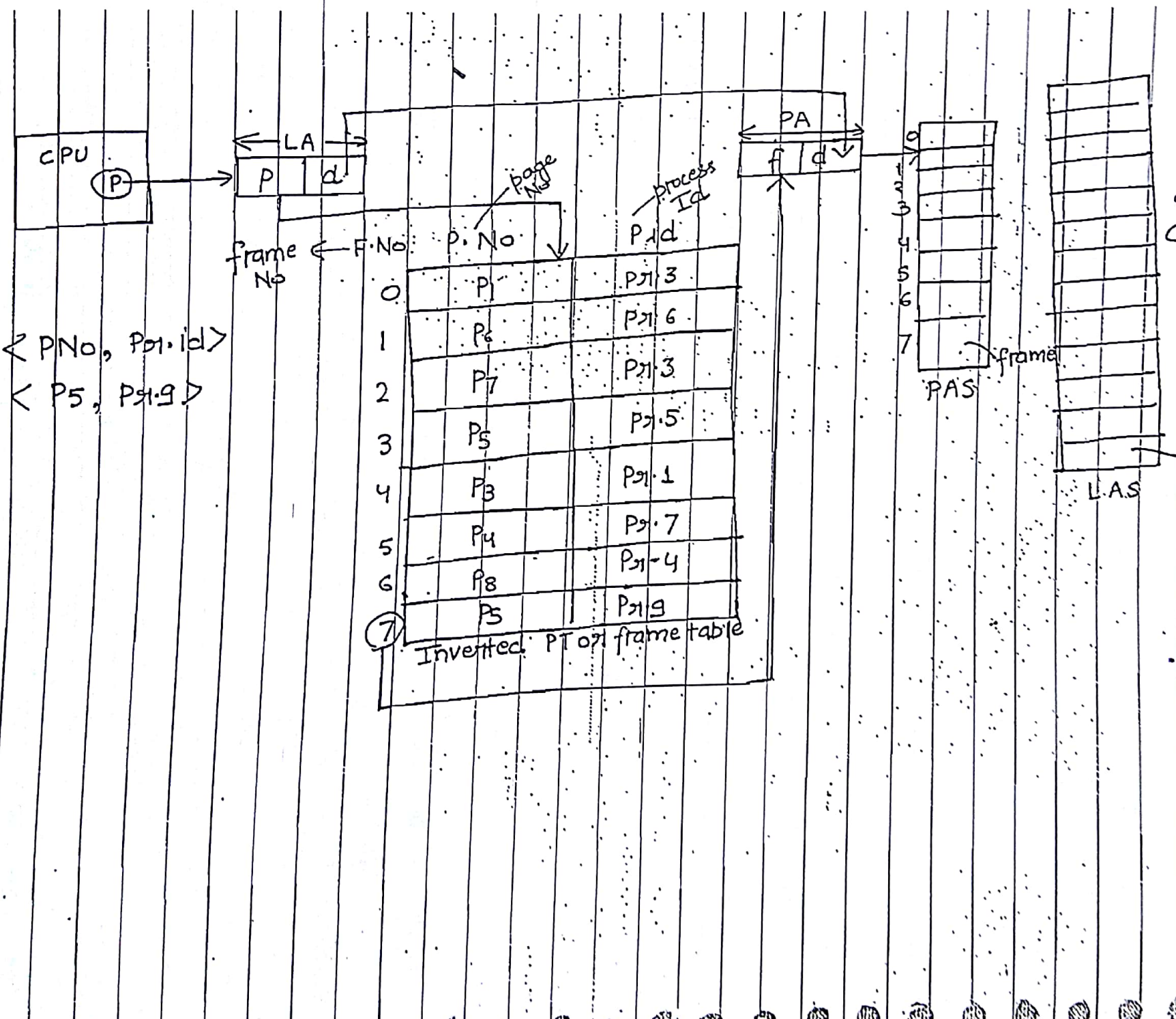
page table size $= \frac{2^{48}}{x}$

II paging - PTS $= \frac{2^{48}}{x^2} \times 4$

III paging $\left[PTS = \frac{2^{48}}{x^3} \times 4\right] \Rightarrow \frac{2^{52}}{x^3} = x$ $\quad x^4 = 2^{52}$

$x = 2^{13}$ B

$= 8$ KB


CPU
P
LA
P
d
PA
f
d
page No
process Id
frame No
F.No
P.No
P1 d
0
1
2
3
4
5
6
7
P1
P6
P7
P5
P3
P4
P8
P5
P1 3
P1 6
P1 3
P1 5
P1 1
P1 7
P1 4
P1 9
< PNo, P1 Id >
< P5, P1 9 >
Inverted PT or frame table
PAS
frame
LAS

Date : / /
Page No.

1. To avoid the overhead of maintaining the page table for every process, the inverted page table will be implemented.
2. In the inverted paging, only one page table will be maintained for all the processes.

- No of entries in the inverted page table is same as No of frame in P.A.S.

- page No is taken from logical Address then page No is search along with its process id from page table & then its corresponding frame No is taken.

$\langle P_{no}, Pr.id \rangle$

$\langle P_5, Pr.9 \rangle \Rightarrow 7$ (frame No)

Note- the memory required to maintain page tables of the processes will be less but the searching time for the corresponding page of a process will be more.

Ques - Consider a system which has LA of 34 bits & PA of 29 bits and Page size is 16 KB & page table entry size is 8B. Calculate
(i) Conventional Page table size.
(ii) Inverted Page table size.

$$\text{No of frames} = \frac{2^{29}}{2^{14}} = 2^{15}$$

$$\text{No of pages} = \frac{2^{34}}{2^{14}} = 2^{20}$$

$\therefore$ (i) Con. page T size $= 2^{20} \times 8B = 8MB$

(ii) Inverted PT size $= 2^{15} \times 8B = 256\,KB$

Date: / /
Page No.

23 Ratio = $\frac{2^{32}/2^{12}}{2^{17}/2^{12}} = 2^{15} : 1$

5. LA = 32 PA = 30
page size = 4kB
No of pages = 2^{20}
No of frames = 2^{18}
size = $2^{18}(20+12) = 2^{20} \times 8$
$= 2^{20}$ B

Segmentation-

Ex - d = 210 for segment 3
d < l. 210 < 200 - false
it will generate trap to OS that it is trying to access the outside value. then word No is gen
if d < l then particular executed of that segment
d = 190
l = 200.
5880 + 190
= 5990 is fetched

- basically user may like diff modules in a program are stored separately at diffr places but in paging they are placed continous.

Date: / /
Page No.

1. Paging does not follows the user's view of memory allocation.
2. To achieve the users the view of memory allocation the segmentation will be implemented.
3. In the segmentation, LAS will be divided into various segmer
4. the ~~si~~ segment of LAS will vary in the size.
5. the segment of LAS are brought into PAS

s - No of bits required to represent segments of LAS or segment No.
d - No of bits required to represent the segment size or word No. of segment or segment offset.
l - size of segment

$d = max\ segment\ size$

- frame contain segment

- No of entries in segment table is same as No of segments in LAS
- Size of segment is also called as limit of segment
- if $l < d$ ie $d > l$ than CPU is trying to access the word outside the segment so it will generate a trap to OS.
- ~~d+l=w~~ B.A+d will generate the word No to be accessed, where instⁿ are actually.

- Since segments are of variable size, when the variable size the segments brought from L.A.S to P.A.S so it is ~~semilar~~ similarly behaving like variable partition scheme. Hence, segmentation still suffers from external fragmentation.

~ the entire segment is plotted to main memory.

Date: / /
Page No.

24

LA		
3	222	
s	d	

physical address = 498 + 222 (222 < 302)
= 720

Segmented Paging - "PAGING ON SEGMENT, WHEN SEGMENT SIZE INCREASES"

X = Base Address of page table of Respective segment

LA = 34 bits
d = <18,16>
Page size of segment = 1KW
Segment size = 2^{19}W
No of segment = 2^{18}
Page table of segment → table to represent pages of segment

Date : / /
Page No.

Segment Size = 2^{16} W = 64 kW

Segment

- Here, entire segment is not bought into main memory bcoz it is wastage of resource so paging is applied on segment & pages is brought is main memory.

No of pages on segment = $\frac{64kW}{1kW} = 64$

1. To avoid the overhead of bringing the large size segment into memory the segmented paging will be implemented.
2. In the segmented paging, paging will be applied on the segment and instead of bringing entire segment in to m/m, the pages of segment will be brought into memory.
3. No of entries in the page table of segment is same as No of pages in the segment.

- Every segment will have its own page table.

P_1 = No of bits required to represent pages of segment or page No. of segment.

d_1 - No of bits required to represent page size of a segment or word No of a page of segment or page offset of segment.

- frames in PAS are storing pages of segment.
- page size of segment is same as Frame size of PAS.

the segment is divided into 1K pages & each page is having 512 entries. the segment No required 13 bits to represent all the segment of LAS. the Memory is word addressable & page table entry size is 4W. the frame No requires 18 bits to represent all the frames of PAS. calculate.

(i) Length of logical Address
(ii) Length of physical Address
(iii) PTS of segment

Ans- (i) Segment Size - $2^{10} \times 2^9$ $\Rightarrow$ d=19

∴ Logical Address = 32

(ii) Physical address | 18 | 9 | = 27

(iii) PTS of segment - ~~$2^9 \times 4W = 2KW$~~

$2^{10} \times 4W = 4KW$

Ques- Consider a system using segmented paging architecture. the segment is divided into 8K pages & each page is having 2K entries the memory is byte addressable & the segment no. requires 15 bits, to represent all the segments of L.A.S. the PAS is 512 KB & page table entry size is 8 bits than calculate

1. Length of LA | 15 | d |
2. Length of physical address | 13 | 11 |
3. Page table size of segment
4. No of frames in ~~pas~~ PAS

Ans (i) LA = 39 bit

(ii) PA = 19 bit

(iii) PTS of segment = $2^{13} \times 8$ bits

= 8 kB

(iv) No of frame = ~~8~~ 64 $8 \rightarrow 2^8$ 256

345
256
768+64+5
937

Date : / /
Page No.

28. PAS = LAS = 2^{16} B
No of segment = 8
PTE = 2B
page size = ?
page table size = one page
$\frac{2^{13} \times 2B}{x} = x$
$x^2 = 2^{14} \Rightarrow x = 2^7$
$x = 2^7$ bytes

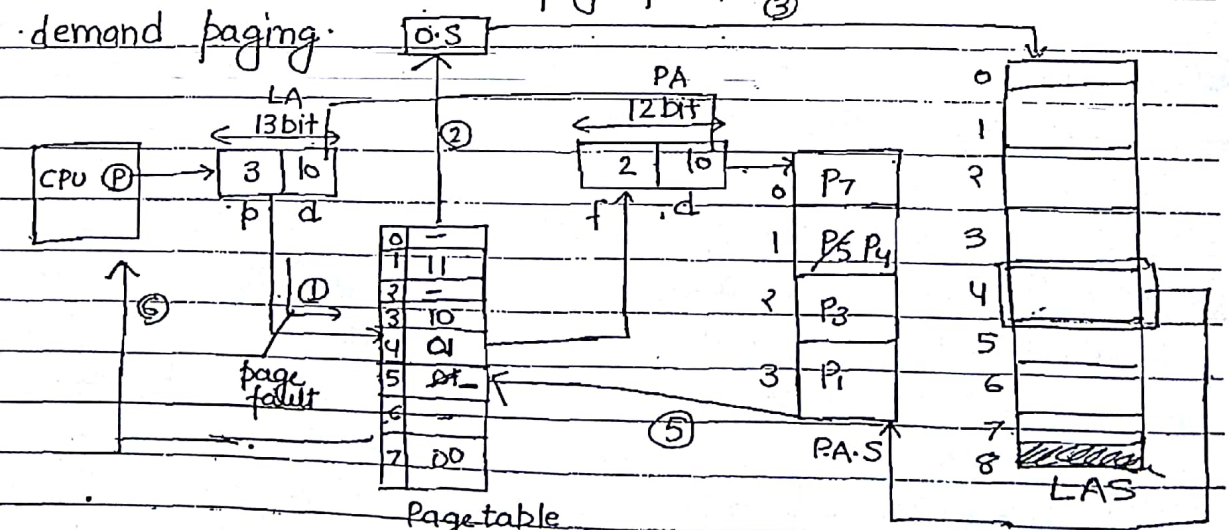
∴ 1 segment size = $\frac{2^{16}}{2^3}$
$= 2^{13}$
No of pages in power of 2

22-Oct-2016 Virtual Memory

- Virtual M/M gives the illusion to the programmer, that the program of larger size than actual physical memory can be executed.
- Virtual M/M is implemented by using demand paging or demand segmentation.

Virtual Memory part of secondary Memory.

Demand Paging - Loading the page into memory on demand (whenever page fault occurs), is called as demand paging.



Since demand of process is always more
of multiprogramming is applied to the system. Less space in physical M/M is provided to the process bcoz more No of processes has to be serviced at a point of time.

Hence, LAS > PAS for a process.

(ii) CPU always generate LA.

• CPU always generate LA bcoz whenever CPU has to execute the process, it has to executed completely. So the whole process is present in LAS, that's why CPU generate the logical address.

If CPU generate PA, the whole process cannot be served as PA < LA.

window - page size = 4KB ?

Date: / /
Page No.

Step6 - i) the signal will be sent to CPU to continue program execution. (by OS)

ii) the CPU will access the required page in the main m/m & continue the program execution.

Initially process is created & ~~stored~~ paging is applied to it these pages get stored to LAS. Some part of it get loaded to PAS CPU generate LA & search for pages in PAS & if pages are present execution continue but if absent than, page fault is managed.

Ex- Window implement the demand paging.

Page Faust Service Time (P.F.S.T) - the time taken to service the page faust is called as page faust service time.

the page faust service time includes the time to perform all the above six steps.

Page Fault Service Time = S
Main M/M access Time = M
page faiilt Rate = p

than the formula for effective M/M access time is

$$E.M.A.T = p*S + (1-p)M$$

$S >> M$

Here page table access time is ignored.

- Service time is quite more as compared to Main M/M access time bcoz in service we are accessing memory more time.

EMAT = 0.25 x 175 + 0.75 x 35

= 43.75 + 26.25

= 70 nsec

Pg 104 20 EMAT = ... S = i + j

$\frac{1}{k}(i+j) + (1-\frac{1}{k})(i)$ M = i

$\frac{1}{k}i + \frac{1}{k} + i - \frac{1}{k}$ p = 1/k ... 1/k 1-1/k

... = $i + \frac{1}{k}$

37 Pg 106 - S = 10 ms

M = 20 ns

page fault = $1/10^6$

EMAT = $\frac{10}{10^6} + \left(1 - \frac{1}{10^6}\right) 20$ ns

EMAT = 30 ns

Pg 103 13 2 ms = p x 0.7 x 20 + p x 0.3 x 8 + (1-p) 1

= 14p + 2.4p + 1 - p

= 16.4p - p + 1

= 15.4p = 1 ms

$p = 1/15.4 = 0.064$

Date : / /
Page No.

1. Whenever page replacement happens in the system two case arises -

(i) when a modified page has to be replaced - in this page fault service time will be more because page getting replaced has to be stored in LAS first & then new page is loaded to PAS.

(ii) when a non-modified page has to be replaced - just overwriting of the page is done.

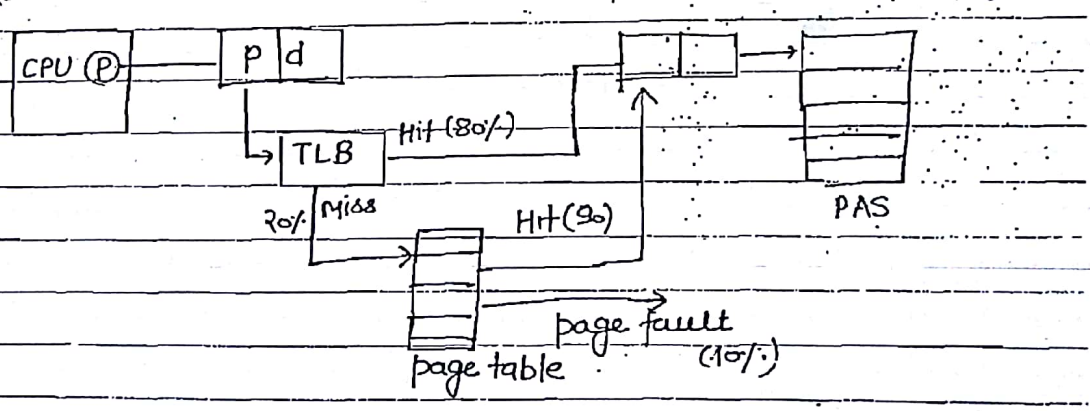
104 Q21 - S = 200 ms M = 10 ms TLB = 80%.

EMAT = 0.8 x 10 ms + 0.2 x 0.9 x 10 + 0.2 x 0.1 x 200

= 8 ms + 1.8 ms + 4

= 13.8 ms

0.18
0.98
9.8
0.2 x 0.1 x 200
0.02



EMAT = 0.8 (0 + 10 ms) + 0.20 (0 + 0.10 x 200 + 0.9 x 10 ms)

↓ TLB hit
↓ TLB access time
↓ M/M access time
↓ TLB miss
↓ page fault
↓ No page fault

Date : / /
Page No.

Ques - Assume the amount of memory on a system is inversely proportional to page fault rate. Each time when M/M doubled, page fault rate is reduced to third (P is made P/3) currently the system has 32MB of memory & the page fault rate is 2% the main memory access time is 500ns And the Effective M/M access time is 300 μsec. if the M/M is increased to 128 MB what is: EMAT

$P \propto \frac{1}{M}$

$P = \frac{\text{constant}}{M}$ $\quad PM = \text{constant}$

$\frac{2M}{3} \cdot P = \text{constant}$

for Main M/M- 32 — 2%

$\frac{2}{3}\text{constant} = \text{constant}$

64 → 2/3 %

128 → $(2/3)^2$ %

$300\mu sec = 0.02 \times S + 0.98 \times 500 nsec$

$= 0.02 \times S + 490 nsec$

$300 - 0.490 = 0.020 * S$

$S = 14975.5\ \mu sec$

$\therefore$ EMAT = 33.24561 + 0.49889

= 33.74 μsec

Date : / /
Page No.

Page Replacement Algorithms:-

Reference string:- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Reference string means page No referred by the process in the logical address.

NOTE:- No of frames allocated to the process is decided by the instruction set architecture.

First In First Out (FIFO) Algorithm:- Assume 4 frames are allocated to the process
- Initially Empty

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

∴ No of page fault = 10

for 3 frames

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

{No of page fault = 15}

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

Assume (i) 3 frame

(ii) 4 frame

(i)

	1	2	3	4	1	2	5	1	2	3	4	5
			3	3	3	2	2	2	2	2	4	4
		2	2	2	1	1	1	1	1	3	3	3
	1	1	1	4	4	4	5	5	5	5	5	5
	F	F	F	F	F	F	F			F	F	

No of fault = 9

(ii)

	1	2	3	4	1	2	5	1	2	3	4	5
				4	4	4	4	4	4	3	3	3
			3	3	3	3	3	3	2	2	2	2
		2	2	2	2	2	2	1	1	1	1	5
	1	1	1	1	1	1	5	5	5	5	4	4
	F	F	F	F			F			F	F	F

No of fault = 10

Note- Belady's Anomaly - By increasing No of frames to the process the No of page fault should decrease but instead they are increasing, this problem is called as - Belady's Anomaly.

(ii) Optimal page Replacement - In the event of page fault, replace the page which will not be used for longest duration of time, in future.

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

Assume frame = 4

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

FIFO FIFO

Date : / /
Page No.

No of page fault = 8

Assume frame = 3

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

No of page fault = 9

LRU Page Replacement- In the event of page fault, replace the page which is least recently used.

Ref. String - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Assume frames = 3

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

No of page fault = 12

Assume frame = 4

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

No of page fault = 8

Date: / /
Page No.

(iv) MRU (Most Recently Used) page replacement - In the event of page fault, replace the page which is most recently used.

Ref string - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Assume frames = 4

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

{ Total fault = 12 }

Assume frame = 3

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

{ Total faults = 16 }

104 17 ⇒ 1, 2, 4, 4, 5, 5, 5, 1, 2, 2, 2, 3, 3

address - 0100, 0200, 0400, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0370

page size = 100 records

Date : / /
Page No.

Record No	page No
0 - 99	0
100 - 199	1
200 - 299	2
300 - 399	3
400 - 499	4
500 - 599	5

Reference String
1, 2, 4, 4, 5, 5, 5, 1, 2, 2, 2, 3, 3

for frame = 1

∴ No of page faiut = 7

Since Reference String never contain continous same pages.
1, 2, 4, 5, 1, 2, 3
∴ final Reference string- 1, 2, 4, 5, 1, 2, 3
No of frames = 1
∴ Total page fault = 7

1	2	4	5	1	2	3
F	F	F	F	F	F	F

NOTE- The reference string will never contain continuous same page.
Eg: 2, 3, 4, 1, 1, 6, 2 → invalid
2, 3, 4, 1, 6, 2 → valid

12 Pg no 3 (c)

28 0, 0, 0, 1, 1, 2, 2, 0, 4, 4, 5, 5, 1, 2, 5, 5
Ref String- 0, 1, 2, 0, 4, 5, 1, 2, 5 (b)

				4	4	4	4	4
		2	2	2	2	1	1	1
	1	1	1	1	5	5	5	5
0	0	0	0	0	0	0	2	2
F	F	F		F	F	F	F	

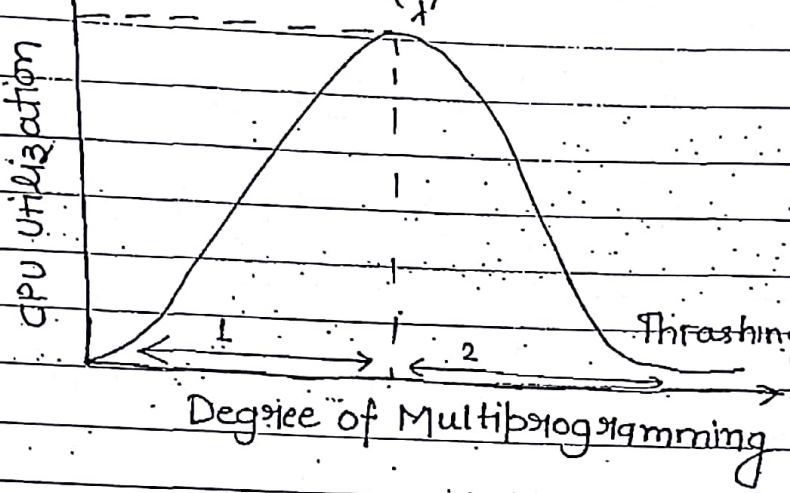
24. (a) 28.

				3	3	3	3	3

25.

			3	3	3	5	5	
			2	2	2	2	6	

(c)



When No of processes increases in main m/m, till a point CPU utilization increases but afterward it decreases because of less resources.

Total No of frame ⇒ 400

No of process = 100
Each process P_i gets = 4 frames

Increase ↳

No of process = 400
Each process P_i gets = 1 frame

In the initial degree of multiprogramming, upto the point of 1, the CPU utilization is very high & the system resources are utilized perfectly - If we further increase the degree of multiprogramming, the CPU utilization will drastically fall down & system will spend more time only in the page replacement & the time taken to complete the exen of process will increase, this situation in system is called as thrashing

Date : / /
Page No.

Causes of thrashing -

1. High Degree of Multiprogramming.
2. Lack of frames (Main M/M).

Recovery of Thrashing -

1. Donot allow the system to go into thrashing & instruct the long Term Scheduler not to bring the processes into the Memory after the point of 1.

2. If the system is already in the thrashing than instruct the mid term scheduler to suspend some of the processes so that we can recover the system from the thrashing.

Ques- Consider a system with the following parameters:
1. CPU utilization = 10%
2. paging disk = 90%
3. than which of the following factors will improve the CPU utilization?

(i) Install faster CPU → No bcoz problem is with main m/m
(ii) Install Bigger disk - No
(iii) Increase degree of multiprogramming - No
(iv) Decrease Degree of multiprogramming - ~~No~~ Yes (decreasing will create some spa
(v) Install more main m/m - Yes
(vi) Decrease page size - No (as No of page fault increases)
(vii) Increase page size - Likely

Paging Disk - time spend on page replacement

Date : / /
Page No.

File System-

File

Def^{n}: collection of logically related entities (record)

File will have various attributes:

(i) Name,
(ii) Type,
(iii) location,
(iv) Size,
(v) Owner,
(vi) Date of Creation,
(vii) Last Modification Date,
(viii) Password etc

⇒ file context

- All the attributes of file is called as file context.
- File context will be stored in the FCBC file control Block).

Types-

I. .doc	VI .pdf		XI .mp4
II. .txt	VII. .bat		XII .mp4
III. .obj	VIII. .xslx		XIII .mkv
IV .class	IX: .c|.cpp|.java|.php|.xml|.html		
V .exe	X. .mp3		XIV . flv
XV .apx	XVI .png (paint)		

Various operations will be performed on the file-

1. Create()	5. Append	9. close	13. Rename
2. open()	6. truncate	10. copy	14. Save
3. Write	7. delete	11. paste	15. save as
4. Read	8. hide	12. cut	16. send
17. Undo	18. print		

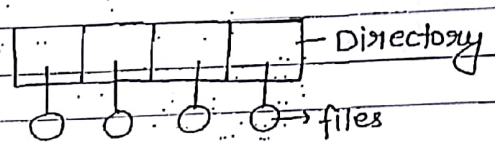
Date : / /
Page No.

File is having various access Methods-
(i) Sequential Access.
(ii) Random Access.

For the better classification of file, file will be stored in the directory.

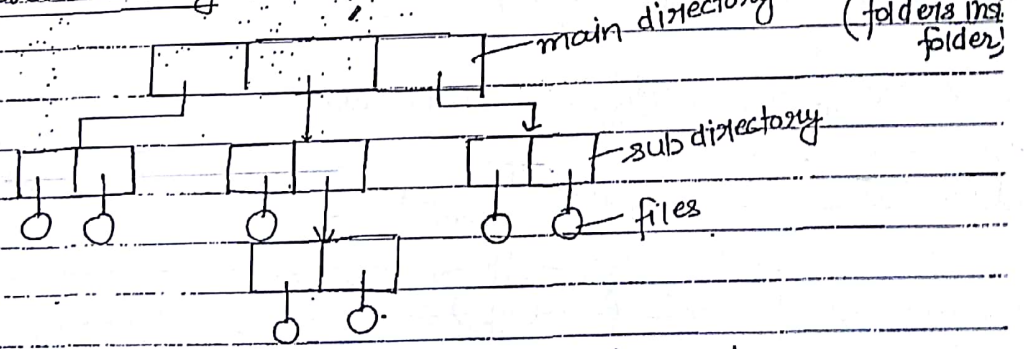
- Directory is similar to a folder.

Directory structure-
(i) Single Level Directory-



1. the implementation of directory structure is easy & simple.
2. the two files cannot have same name.
3. Searching time for the specific file will be more.

To avoid the problem of searching time, we'll move towards multilevel directory-

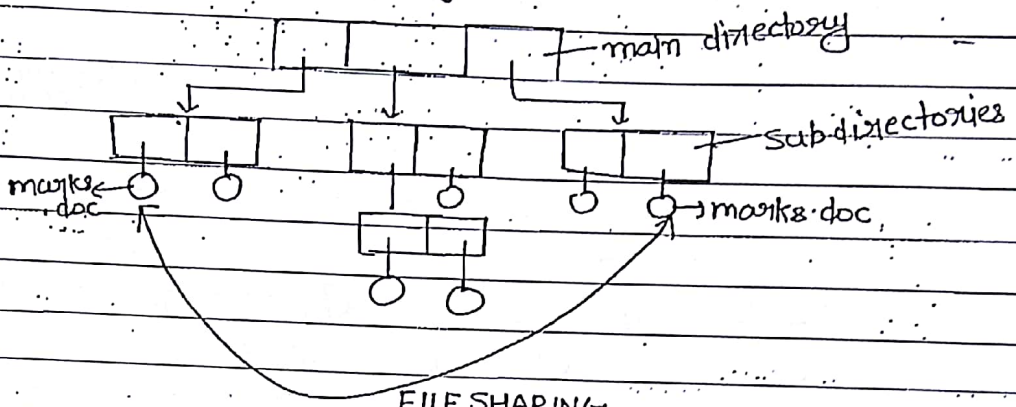


1. The implementation of structure is complicated.
2. the better classification of the files as per the criteria.
3. Searching time for the specific file will be less.
4. If the same file exist in two different directories (sub), if one file is updated, other file has to be updated accordingly. otherwise, there will be an inconsistency.

Date : / /
Page No.

To avoid the problem of inconsistency, we go for Acyclic Graph Directory.

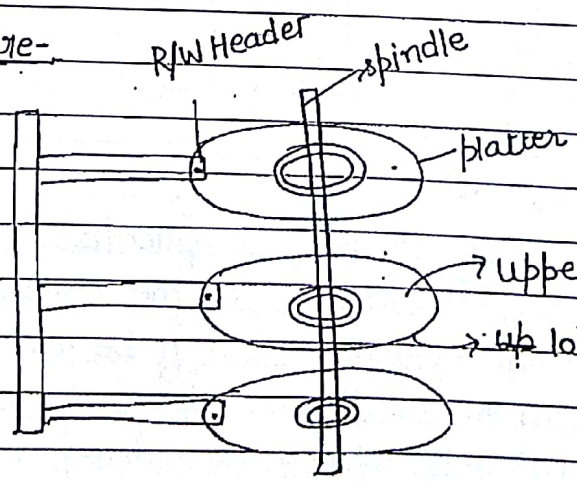
Acyclic Graph Directory-



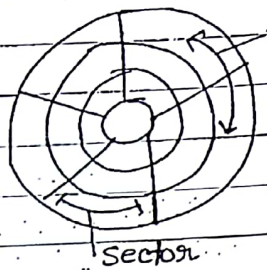
1. the implementation of the directory structure is complicated.
2. the better classification of the files as per the criteria.
3. Searching time for specific time will be less.
4. If the same file exist in the two different directories, than if one file is updated than other file will be updated automatically by using file sharing concept.

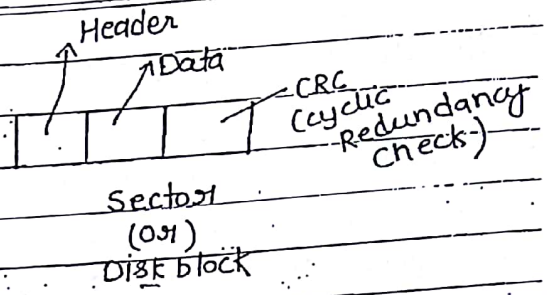
Files are getting stored in to Hard disk.

Disk Structure-



Date : / /
Page No.





Ques- Consider the disk which has 16 platters & each platter is having two surface & each surface is divided into 1k track & each track is further divided into 512 sector. Each sector can store 2kB data.

Calculate-

(i) Capacity of the disk $= 2 \times 16 \times 1K \times 2KB \times 512 = 32\,GB$

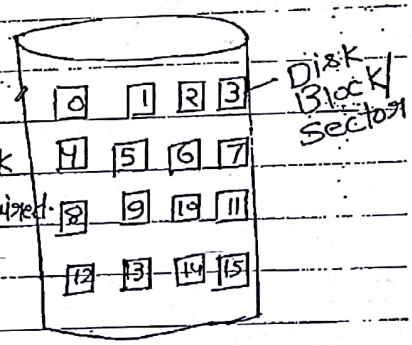
(ii) How many bits are required to identify specific sector of the disk? 24 bits

Disk Space Allocation Methods-

1. Contiguous Allocation- D.B.A - Disk block Address

File	Starting D.B.A	Size
abc.doc	2	4
xyz.doc	9	5

Size - No of disk block required



1. In the contiguous allocation, whenever the file is created, the disk blocks will be allocated in a continuous manner.
2. Increasing the file size may not be always possible. bcoz for increasing the size the next block must be empty but it may or may not be empty.
3. It is suffering from external fragmentation.

Date : / /
Page No.

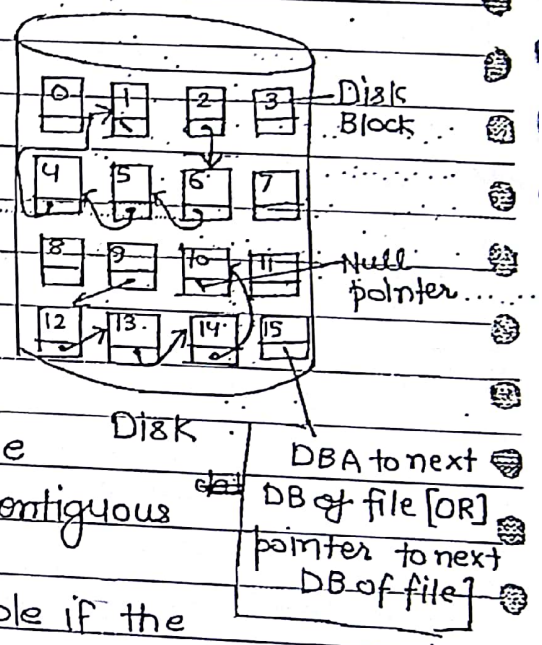
4. the internal fragmentation may exist in the last disk block of the file.
5. It supports both sequential & Random access of the file.

for Random Access = Starting D.B.A + offset	offset - it tells about which block has to be visited of a particular process.

for ppto file, xyz.doc, offset lies b/w 0 to 4.
for visiting of 5th block of xyz.doc
= 9 + 4 = 13 (any block of any file can be visited directly)

2. Linked Allocation (Non-contiguous) Allocation-

file	Starting D.B.A	Ending D.B.A
abc.doc	2	1
xyz.doc	9	10



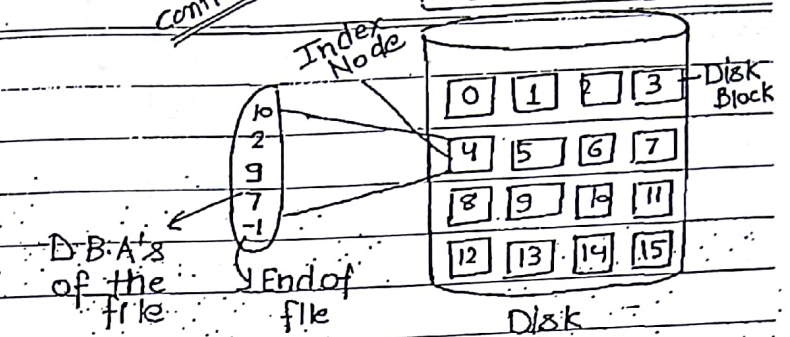
1. In this, whenever the file is created the disk blocks will be allocated in a non contiguous manner.
2. Increasing the file size is always possible if the free disk block is available.
3. There is no external fragmentation.
4. Internal fragmentation may exist in the last disk block of the file.
5. there is overhead of maintaining the pointer in every disk block.
6. If the pointer of any disk block is lost, file will be truncated
7. It supports only sequential access of the file.

To avoid problem of overhead & pointer loss, file obtained Indexed Allo

Date : / /
Page No.

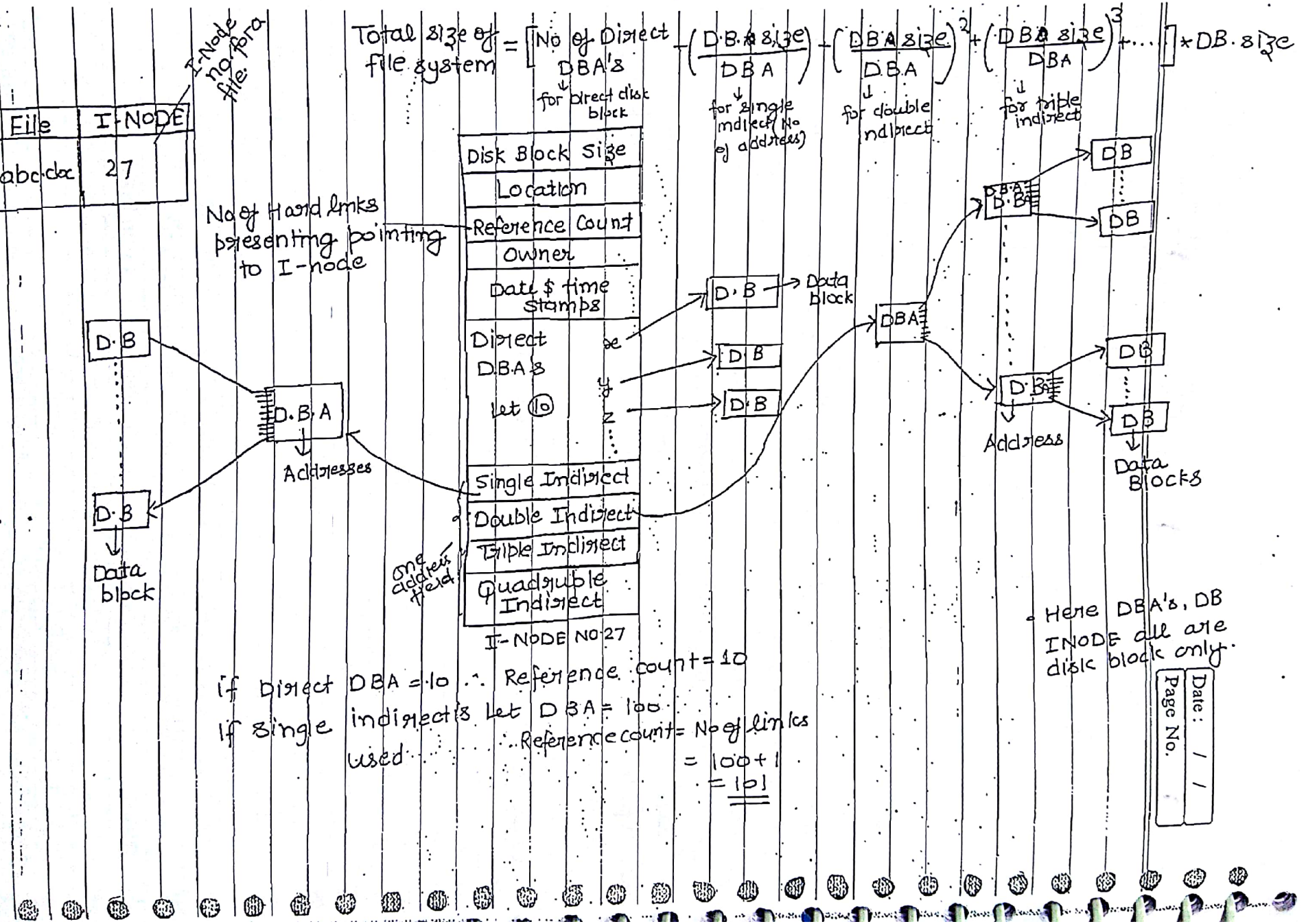
Indexed Allocation-

File	Indexed Node
abc.doc	4



1. In index allocation, every file is associated with its own index node.
2. the index node contains all the disk block addresses of the file.
3. If the file is very large, than one disk block may not be sufficient to store all the disk block addresses of the file. then, more than one block is required & linking is done to them. So, it will again implement the concept of linked allocation.
4. If the file is very small, it is waste of using entire A block just to store the disk block address.

Unix I-NODE implementation- (An extension of indexed allocation,


Total size of file system = [No of Direct DBA's] + (DBA size / DBA) + (DBA size / DBA)^2 + (DBA size / DBA)^3 + ...] * DB size
for direct disk block
for single indirect (No of address)
for double indirect
for triple indirect
File
I-NODE
abc.doc
27
I-Node no. for a file
No of Hard links presenting pointing to I-node
Disk Block Size
Location
Reference Count
Owner
Date & time stamps
Direct DBA's
Let 10
Single Indirect
Double Indirect
Triple Indirect
Quadruple Indirect
one address field
I-NODE NO 27
D.B
Data block
D.B.A
Addresses
DBA
Address
DB
Data Blocks
Here DBA's, DB INODE all are disk block only.
Page No.
Date: / /
if Direct DBA = 10 ∴ Reference count = 10
if single indirect is Let DBA = 100
used ∴ Reference count = No of links
= 100 + 1
= 101

Date : / /
Page No.

1. Unix follows I-NODE way of implementation in order to allocate the disk space to the files.
2. Every file is associated with the its own I-NODE.
3. Depending on the size of the file, the file will be stored only in one particular place, may be in the direct disk block address or may be in the single indirect or may be in the double indirect or in triple indirect & so on.

Reference count - link from I-NODE

$\left(\frac{\text{DB size}}{\text{DBA}}\right)$ ⇒ No of disk block addresses one disk block may contain.

for Maximum size possible -

for quadruple - size = $\left(\frac{\text{DB size}}{\text{DBA}}\right)^4 \times \text{DB size}$

triple size = $\left(\frac{\text{DB size}}{\text{DBA}}\right)^3 \times \text{DB size}$

Ques- Consider Unix I-Node maintain 10 Direct DBA's 1 single indirect one double indirect & one triple indirect disk block address the disk block size is 1kB & disk block address requires 32 bits then what is the max. file size size possible?

Max file size = $\left(\frac{1\text{kB}}{32}\right)^3 \times 1\text{kB}$ (for triplets)

$= (2^8)^3 \times 1\text{kB}$

$= 2^{34}\text{B}$

$= 16\text{GB}$

12 Max- $(128)^3 \times 1024$ B

$= 2^{21} \times 1024$ B

$= 2^{31}$ B → 2GB

7. 4TB = $\left(\frac{1kB}{x}\right)^4 \times 1kB$

$2^{45} = \frac{(2^{13})^4}{x^4} \times 2^{13}$

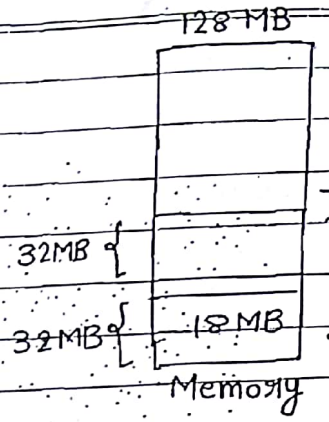
$x^4 = 2^{20}$

$x = 32$ bits

8. 2^{13} 32-13 = 19 bits

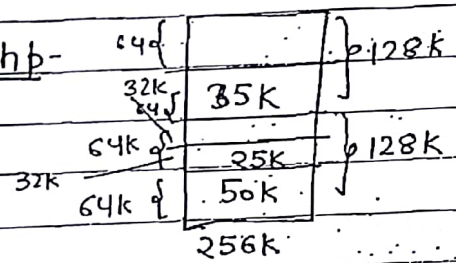
Memory Management Technique-

Date : / /
Page No.



$P_1 \Rightarrow 18$ MB

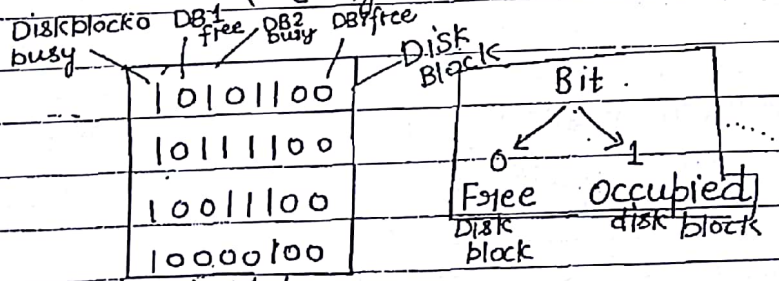
Query chp-



M 9

Bit Map Approach- 1. the bit map approach is used for the disk free space management.

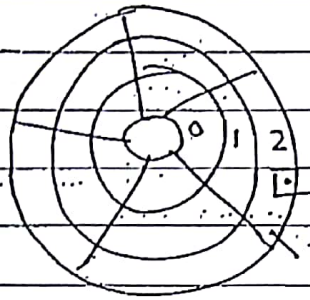
2. Every disk block will be mapped with 1 binary b



free disk block - 1, 3, 6, 7, 9, 14, 15, 17, 18
31, 22, 23, 25, 26, 27, 28, 30,

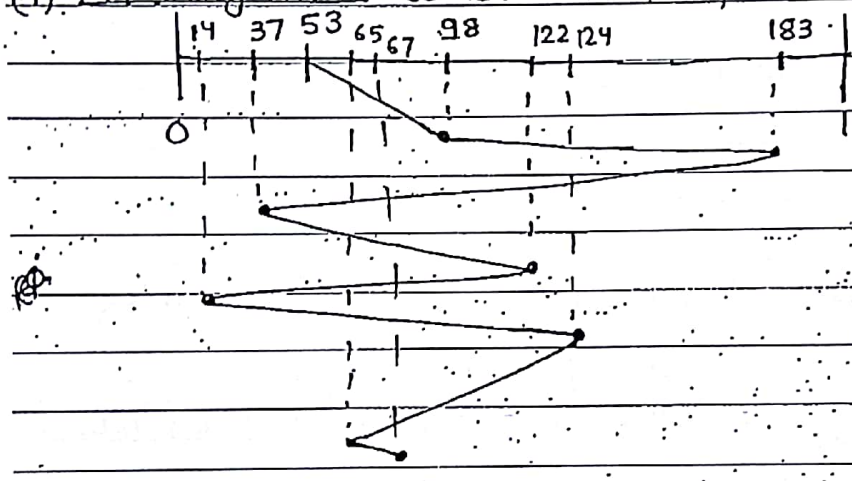
Busy disk block - 0, 2, 4, 5, 8, 10, 11, 12, 13,
16, 19, 20, 21, 24, 29

Track Requests- 98, 183, 37, 122, 14, 124, 65, 67



Goal of Disk Scheduling- to minimize the average seek time of the disk.

(i) FCFS Algorithm- which ever came first will be served first.



No of tracks = 200 (0-199)

present pos^{n} = 53rd track

Total track movements made by R/W Header (Seek time)

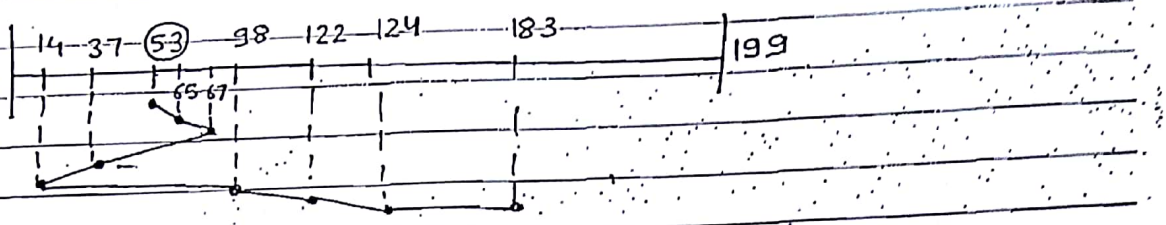
= (183-53) + (183-37) + (122-37) + (122-14)
+ (124-14) + (124-65) + (67-65)

= 130 + 146 + 85 + 108 + 110 + 59 + 2

= 640

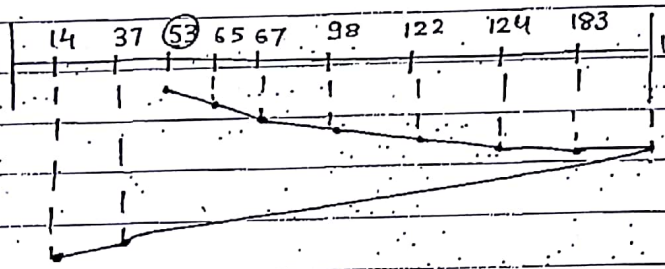
Date: / /
Page No.

Shortest Seek time first (S.S.T.F) 98, 183, 37, 122, 14, 124, 65, 67



total seek time = 12 + 2 + 30 + 23 + ~~84~~ + (183-14)
= 37 + 30 + 169
= 236

3. SCAN (Elevator)- 98, 183, 37, 122, 14, 124, 65, 67

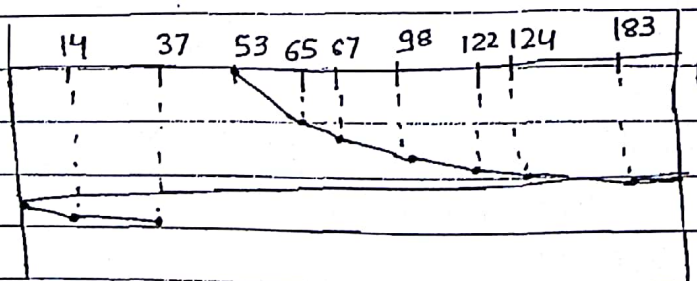


• it will move in a particular direction & go to end of track & then come back.

Seek time = (199-53) + (199-14)
= 146 + 185
= 331

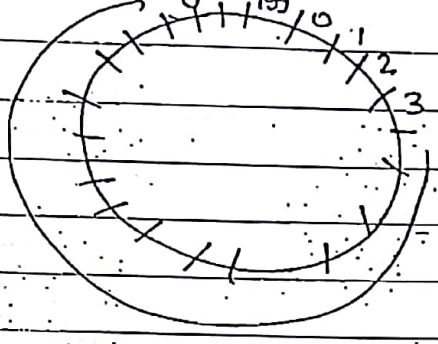
4. C-SCAN: (Circular SCAN)

98, 183, 37, 122, 14, 124, 65, 67 • moving towards right



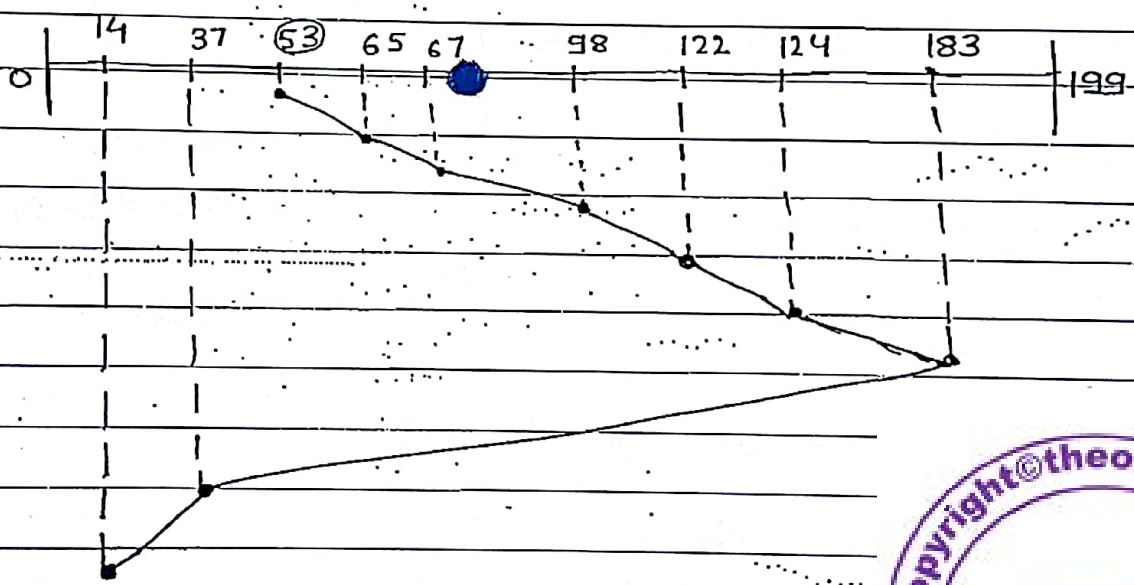
• (199-0) has to be included as it is not practically possible to jmp

Total Seek time = (199-53) + (199-0) + 37
= 146 + 199 + 37



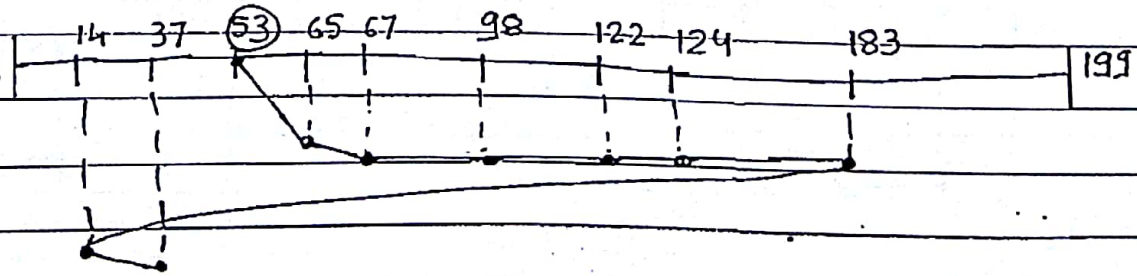
i.e after 199, you can directly jump to zero. It is not possible practically.

5. LOOK: (Look for the last pending request in the direction of read write header) -



Total seek time = (183 - 53) + (183 - 14)
= 130 + 169
= 299

5. C-LOOK: (Circular look)



Total seek time = (183 - 53) + (183 - 14) + (37 - 14)
= 299 + 23
= 322