

O.S

S.No	Topics (SubTopics)	Date	P.no
I.	<p>Introduction & Background</p> <ul style="list-style-type: none">→ Definition→ Goals of operating System.→ Types of operating System<ul style="list-style-type: none">1. Batch operating System2. Multiprogramming OS3. Multitasking OS4. Multiprocessor Systems5. Real Time Systems.	19/1/19	/
II.	<p>PROCESS MANAGEMENT</p> <ul style="list-style-type: none">→ Definition→ Attributes→ states & State Diagram→ CPU bound & I/O bound→ Context Switching→ Three schedulers (Long, medium, short)→ Different times (Arrival time, completion Time)		

S.No	Topics (Subtopics)	Date	P.No
	<u>SYNCHRONIZATION</u>	20/11/19	
	→ The Producer- Consumer → Printer Spooler Daemon → Basic definition (critical section, Race condition) → Conditions required to achieve synchronization. → Solutions. → Lock Variables → Starvation (Dekker's Algorithm) → Petersons Algorithm → TSL Instruction set (Hardware type)	20/11/19	77 79 83 87 93
	<u>SEMAPHORES</u>		94 98
	→ Counting Semaphore → binary Semaphore		
	<u>CLASSICAL PROBLEMS OF IPC</u>		
	→ Producers- Consumers with Semaphore. → Readers - writers	107	112

I. Introduction & Background

II. Process management

- process concept
- CPU scheduling
- synchronization
- concurrent programming
- Deadlocks
- Threads.

III. Memory management

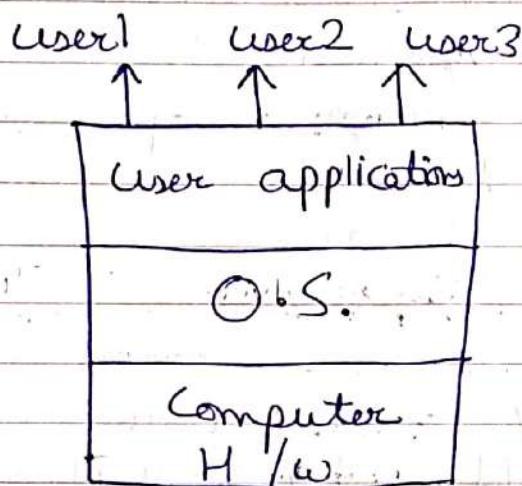
- RAM chip implementation
- Loading, Linking and Address Binding.
- Techniques
 - paging
 - multilevel paging
 - inverted paging
 - Segmentation
 - Segmentation paging
- Virtual memory.

IV. File systems.

INTRODUCTION AND BACKGROUND

Q1.) What is an O.S.?

(1.Ans.) O.S. is an interface between user and computer H/w.

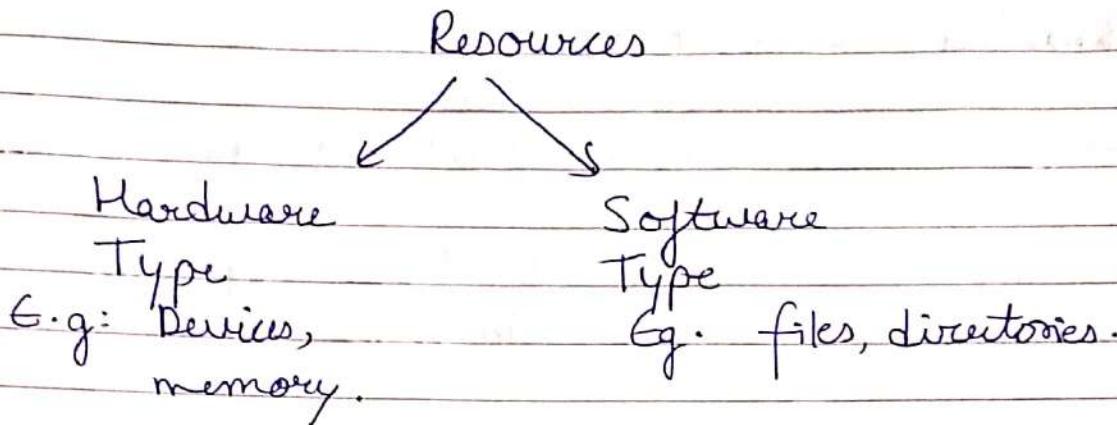


```
main()
{
    int n;
    printf ("Hello");
}
```

printf internally calls write() system call in order to communicate with the monitor.

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

→ OS is also called as resource allocator because it is responsible for allocating resources of a computer.

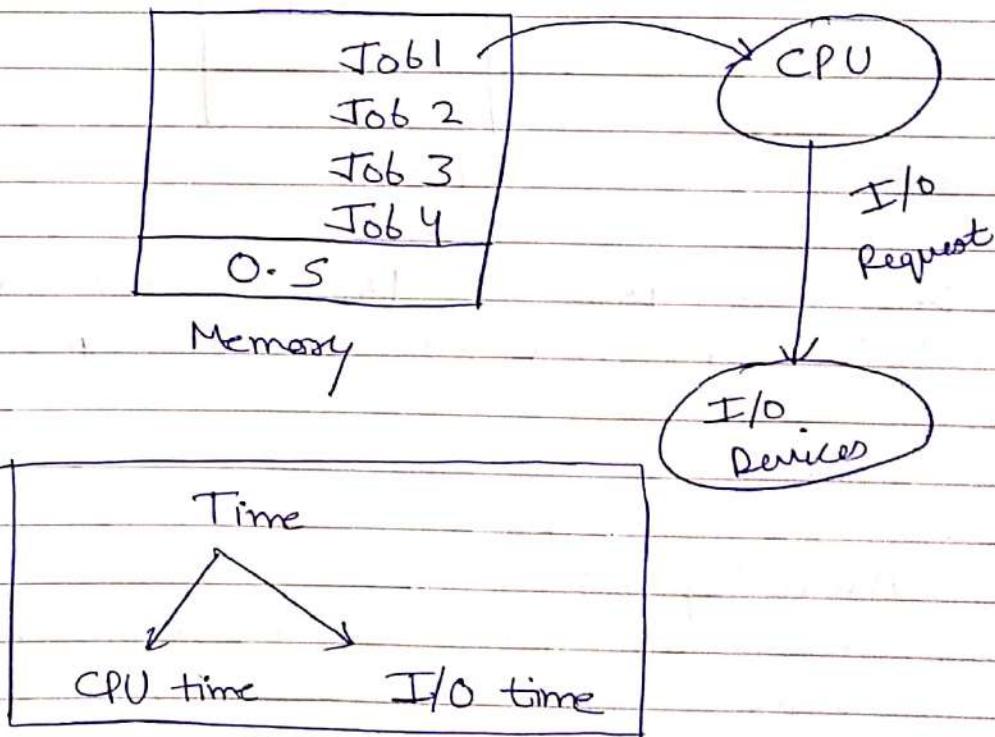


GOALS OF OPERATING SYSTEM:-

- 1) The primary goal is convenience.
- 2) The secondary goal is efficiency.

TYPES OF O.S:-

1.) BATCH OPERATING O.S:-



→ If the job is completed completely, then only another job will be scheduled onto CPU.

Advantages of Batch O.S:-

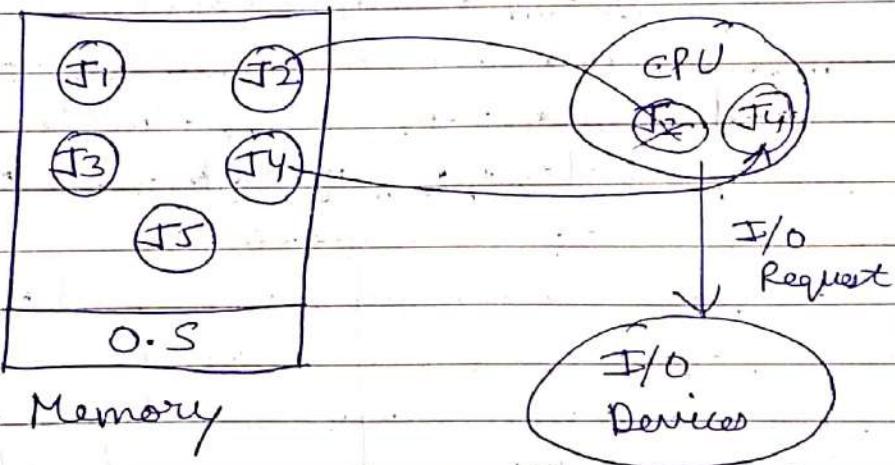
→ Increased CPU idleness.

→ Decreased throughput of the system.

Throughput - The No. of Jobs completed per unit time.

Eg: IBM OS/2.

2.) MULTIPROGRAMMING OS:-



→ If the job is leaving the CPU to perform I/O operation, then another job, which is ready for execution will be scheduled onto the CPU.

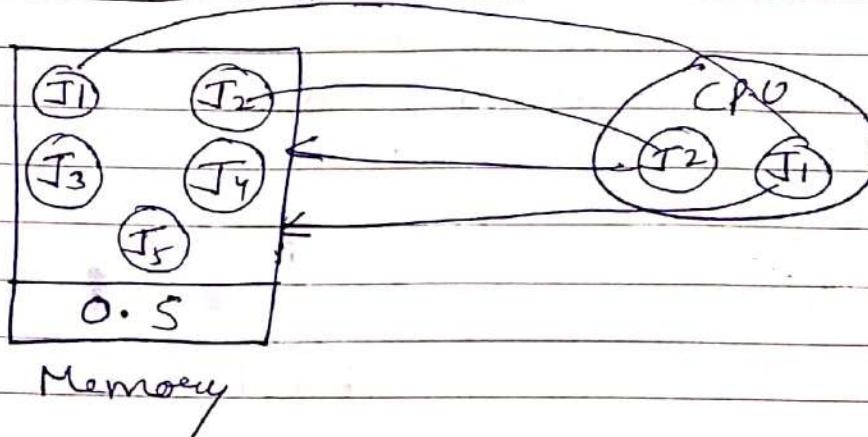
Advantages:-

→ Increased CPU utilization.

→ Increased throughput of the System.

Eg:- Windows, Unix, Linux.

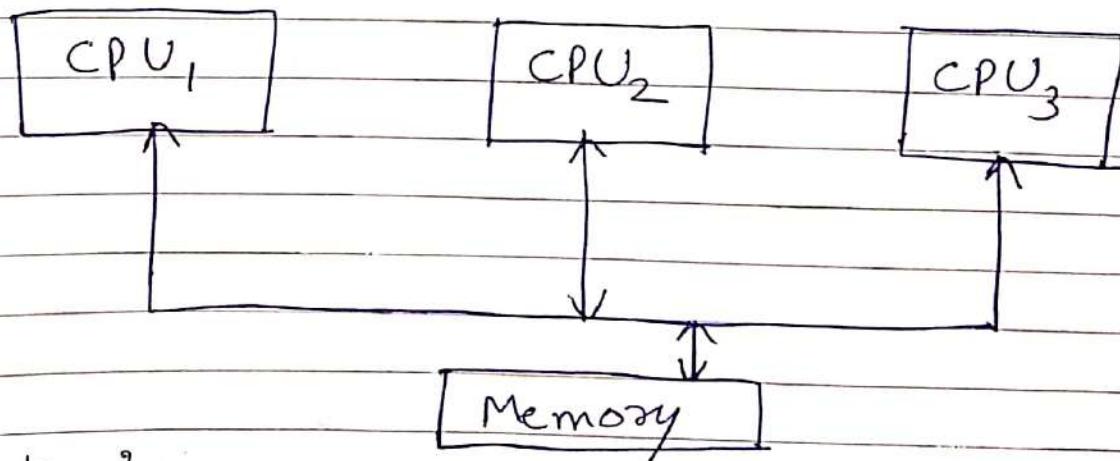
3.) MULTITASKING O.S.:-



- Multitasking is an extension of multiprogramming O.S.
- The jobs will be executed in the time sharing mode.
- At any point of time only one job can be done

Eg :- Windows, Unix.

4.) Multiprocessor Systems:-



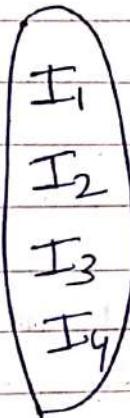
Advantages:-

- Increased throughput of a system.
- Reliability.

↳ fault tolerant systems.

5.) REAL TIME SYSTEMS:-

The Systems which are strictly deadly time based are called real time systems.



Hard real
time

Eg: Satellite system,
missile systems,

soft real
time

Eg:- Banking
sector.

Minor delay is not
acceptable

Minor delay
is acceptable.

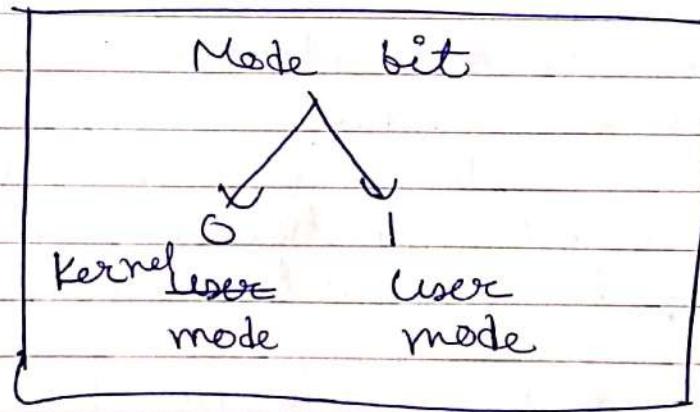
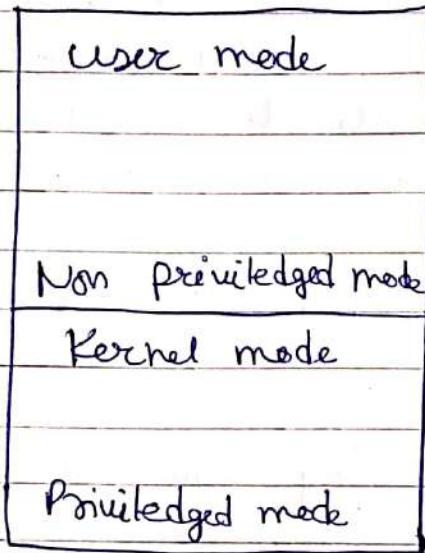
Eg:- S_xwalks, V_xwalks, RTOS.

OPERATION

"errant users"
↓
unauthorised

7

DUAL MODE :-



→ In the hardware level, the instructions are executed by using dual mode operation.

- 1.) User mode/ non privileged mode
- 2.) Kernel mode / privileged mode / System mode / monitor mode.

The dual mode operation is used in order to provide protection and security to the user programs and also to the operating system from errant users.

→ It is purely the decision of the O.S., in which particular mode, the instruction has to be executed. Generally the privileged instructions are executing in the Kernel mode.

→ Non privileged instructions are ~~not~~ executing in the User mode.

→ In the booting time system always starts in the Kernel mode.

→ Operating System always runs only in a Kernel mode.

The mode bit is used to identify in which particular mode, the current instruction is executing.

NOTE:- The mode switching, takes very less time compare to process switching.

(d)

PRIVILEGED INSTRUCTIONS:-

- 1.) I/O operation
- 2.) Context switching
- 3.) Set the time of clock
- 4.) Disabling the interrupts.
- 5.) Changing the memory map.
- 6.) Clearing the memory map.

Non privileged instructions :-

- 1.) Reading the status of the processor.
- 2.) Reading the time of clock.

LAYERED APPROACH:-

Process management	L ₁
Memory management	L ₂
File management	L ₃
Device management	L ₄
Protection & Security	L ₅

O.S

- The design and implementation of the O.S. will be done into multiple layers.
- The main advantage of layered approach is debugging, abstraction and modularity.
- Modularity means if any specific layer is updated, then it will not have any

(g), (d), (a), (c) (d)

impact on other layer.

Fork() System call Implementation:-

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

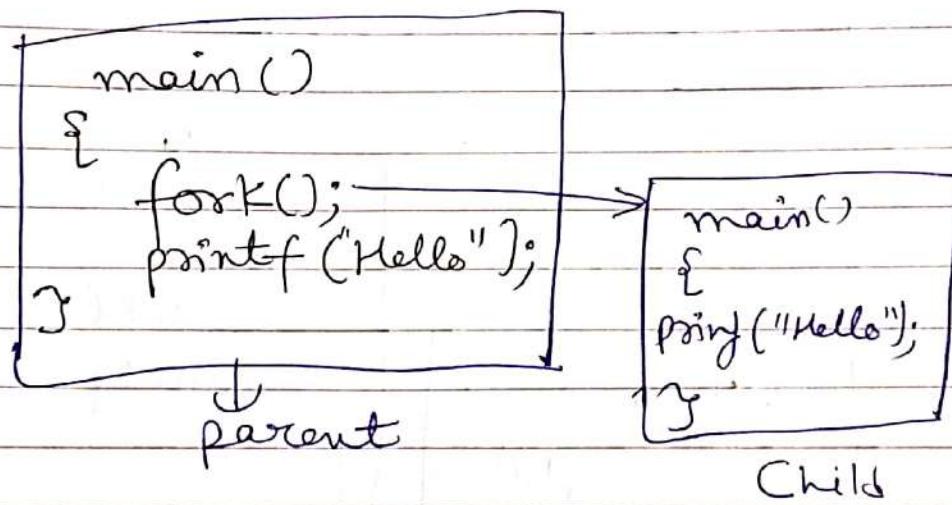
NOTE:- fork is a system call used to create the child process.

- fork returns negative value if the child process creation is unsuccessful.
- fork returns value 0 to the newly created child process.
- "fork returns positive value" to the parent process.

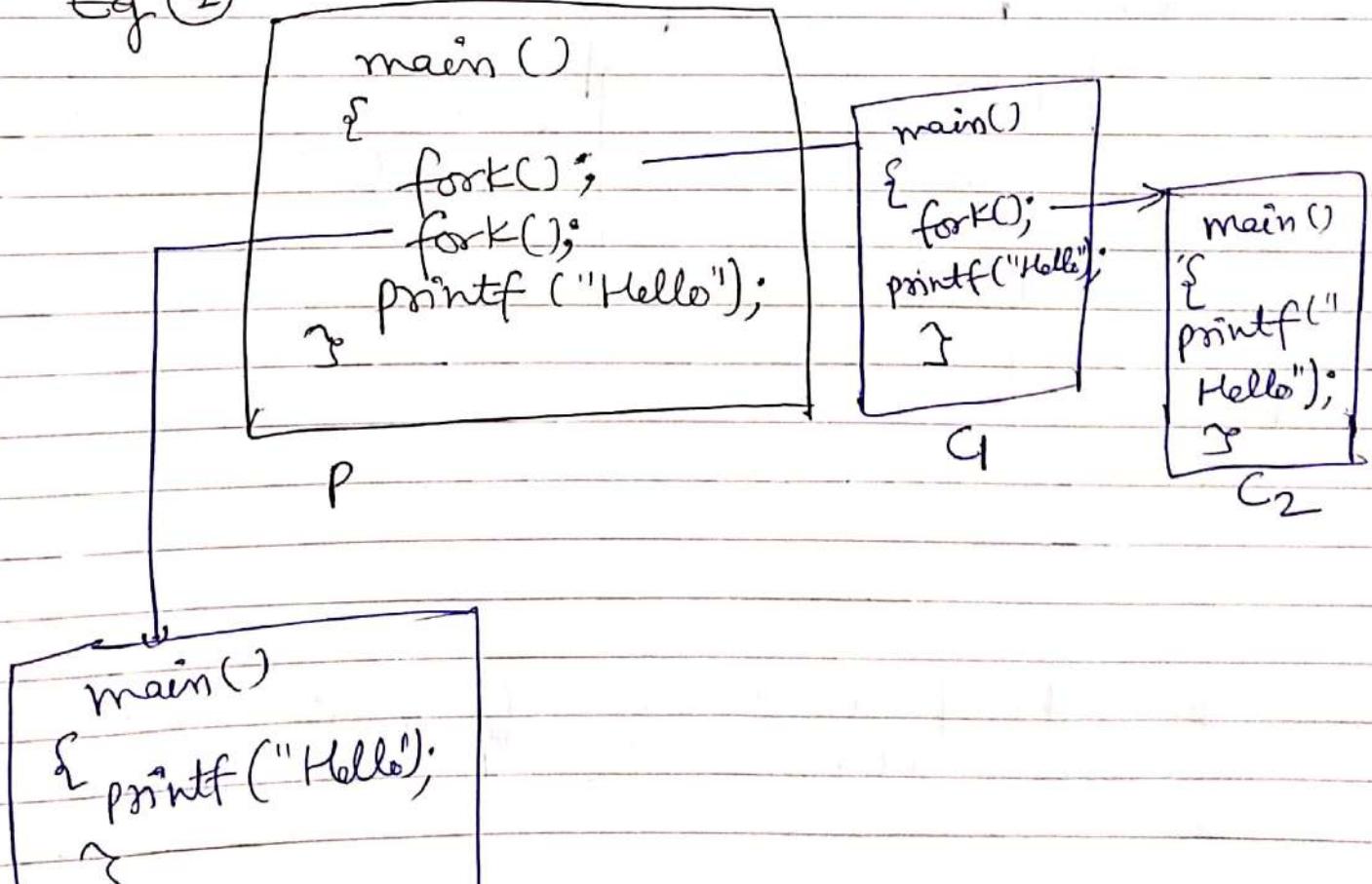
↓
(process id of a child process)

→ When the child process created, then the new memory location will be allocated to the child process and parent and the child process will have same virtual Address; but physical address will be different.

Eg(1) :-



Eg(2)



(eg.)

main ()

{

fork();

fork();

fork();

printf ("Hello");

}

main()

{ fork();

fork();

pf ("Hello");

C₁

main()

{ fork();

pf ("Hello");

C₂

main()

{ pf ("Hello");

main()

{

fork();

printf ("Hello");

}

main()

{ printf ("Hello");

main()

{ pf ("Hello") }

C₃

7

NOTE :- If the program have N fork calls
then it will [2^{N-1}] child process.

PROCESS MANAGEMENT

Definition:- The program under execution is called as process.

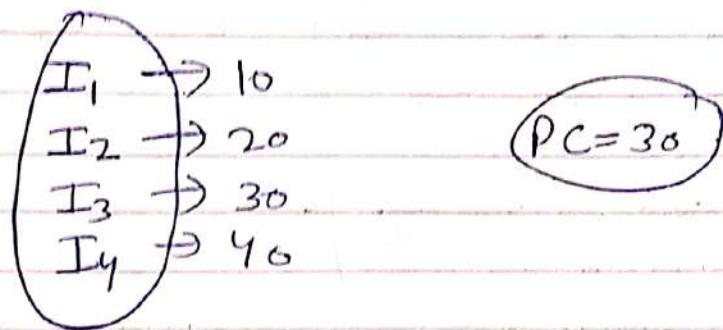
- It should reside in the main memory.
- It occupied the CPU to execute the Instructions.
- Active and Dynamic.
(load on memory)
(taking Inputs and producing outputs)

Attributes:- Process will have various
Attributes:-

- 1.) Process id.
- 2.) Process state.
- 3.) Priority.
- 4.) Program counter.
- 5.) General purpose registers.
- 6.) List of open files.
- 7.) List of open devices.
- 8.) Protection information.

- 1.) Process id:- Process id is unique identification number, which is assigned by the operating system at the time of process creation.
- 2.) Process state:- The process state contains current state information of a process, where it is residing.

- 3.) Priority :- is a parameter assigned by the operation system at the time of process creation
- 4.) Program Counter :- Program counter contains address of the next instruction to be executed.



- * All the Attributes of a process is called as context of a Processor.
- * The context of a process will be stored in a P.C.B (process control block).

P. id	P. S
priority	P. C
G. P. P.	L. O. F.
L. B. D	P. I

P.C.B.

- Every process will have its own P.C.B.
- PCB of the processes will be stored in the main memory.
- PCB will be implemented by using double linked list.

A Process will have various states:-

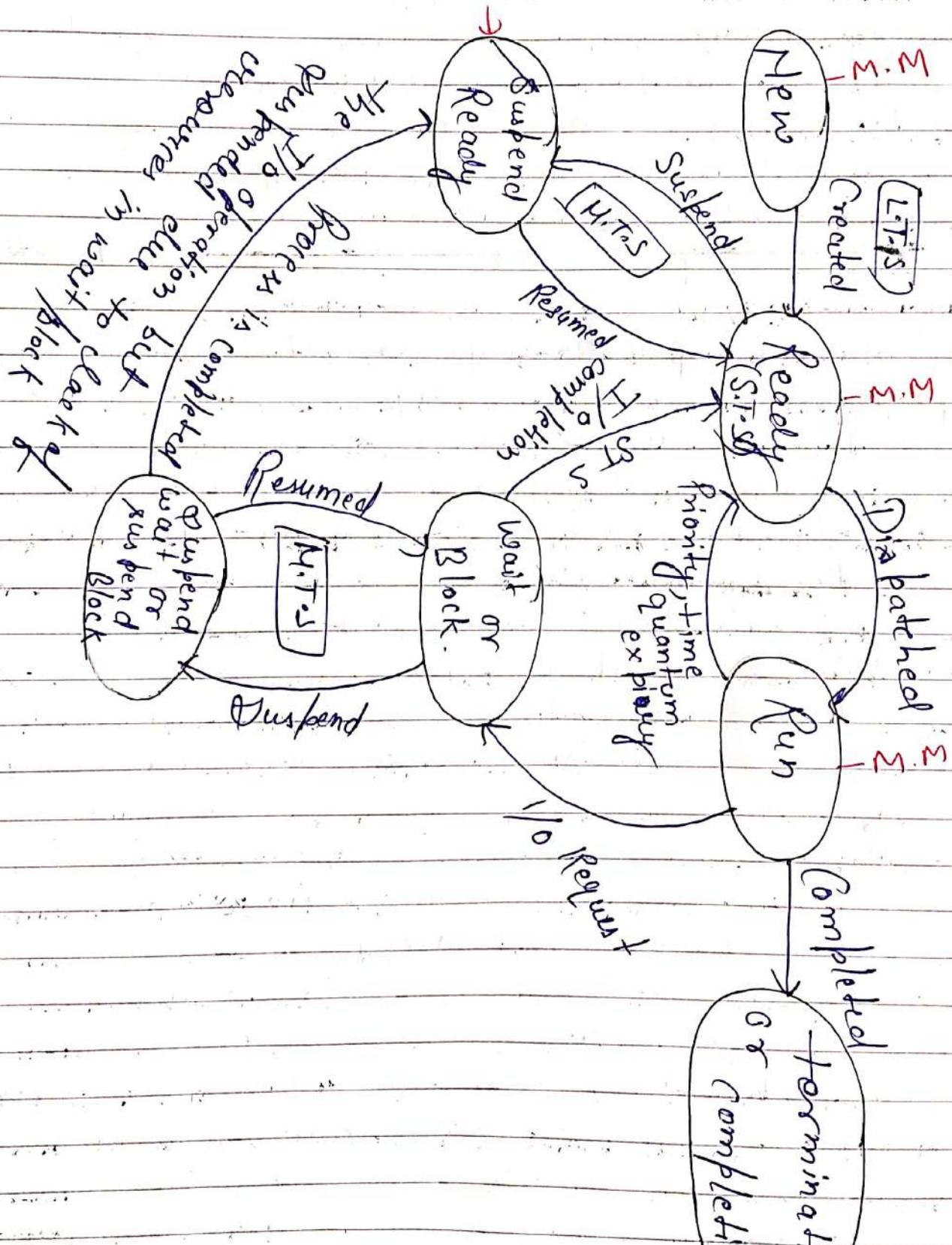
- 1.) New
- 2.) Ready
- 3.) running
- 4.) wait or block
- 5.) termination or completion.
- 6.) Suspend ready
- 7.) Suspend wait or suspend Block.

Various Operations performed on the processor :-

- 1.) Creating
- 2.) scheduling
- 3.) dispatching
- 4.) executing
- 5.) Killing or terminated
- 6.) Suspending.
- 7.) Resuming.

Process State Diagram :-

Banning state



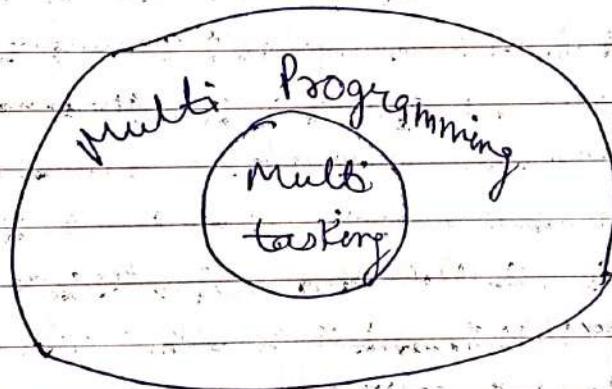
- Initially process will be in the new state, it means process is under creation or process is being created.
- Once the process is created, it will move on to ready state & in ready state there will be multiple no. of processes.
- One of the process will be selected & that will be dispatched onto the running state.
- When the process is the running state, it occupies the CPU & executing the instruction of the process & performing CPU time.
- In the running state, there will be only one process at any point of time.
- If the running process requires I/O operation, then it will be moved on to wait & block state & in wait & block state also their can be multiple process, it means multiple processes will perform I/O operation simultaneously.

Multi Programming :-

→ Non preemptive
(Co-operative scheduling)

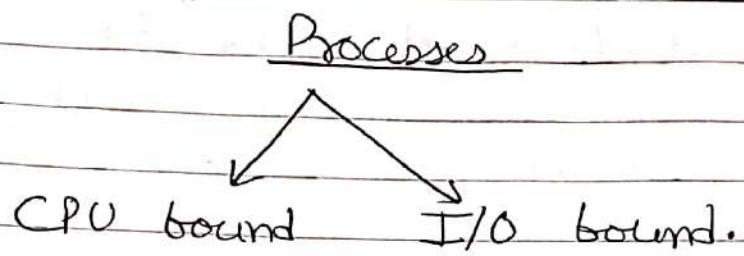
Preemptive
(or)
multitasking
(or)

time sharing (time quantum)



- when the process is in the ready, running & the wait states then it is residing in the main memory.
- If the more no of process are getting created, then the resources may not be sufficient to manage the processes in the ready state. In that case, some of the processes will be suspended & they will be moved onto suspend ready state.

- when the process is in suspend ready state, then it is residing in the



- * CPU bound :- The processes which require more amounts of CPU time are called as CPU bound.

- These process will spend more time in a running state.

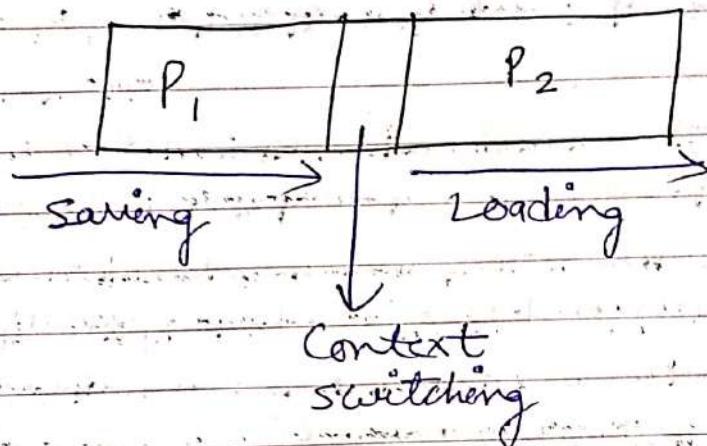
- * I/O bound Process :- The processes which require more amount of I/O time are called as I/O bound proc.

- These process will spend more time in the wait or block state.

DEGREE OF MULTIPROGRAMMING :- The No. of processes present in the main memory at any point of time is called as degree of multiprogramming.

Each and every time, when the process is moving from one state to other state the context of a 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, context switching time will also increase, which is undesirable.

Context switching time is considered as overhead (burden) of the system.

NOTE:- In some special cases, if there is only one process, then still it is considered as a context switching.

3 - Schedulers :-

JOB SCHEDULER

1.) LONG TERM SCHEDULER (L.T.S.):- Long term scheduler is responsible for creating & bringing new processes into the system.

2.) SHORT TERM SCHEDULER (CPU SCHEDULER)

STS:- is responsible for selecting one of the process in the ready state for scheduling onto the CPU.

3.) MID TERM SCHEDULER (MTS):-

(Medium term scheduler) is responsible for suspending & resuming the processes. (swapping)

→ The job done by MTS is called as a swapping:-

DISPATCHER:- is responsible for loading the selected job onto the CPU

→ Dispatcher is also responsible of performing context switching.

→ Long term scheduler should select good combination of both the CPU bound & the I/O bound process in order to get a good throughput for the system.

→ ITS controls the degree of Multipro-

Q1.) Consider a system which has n CPU processors, Then what is the minimum & maximum number of process that may present in the ready, running and wait state?

Ans)

	Min	Max
Ready	0	Any
Run	0	n
wait	0	Any

Any depends on maximum degree of Multiprogramming in the system.

Process have various different Times:-

- 1.) Arrival Time :- The time when the process is arrived into ready state is called as arrival time of a process.
- 2.) Completion Time :- The time when the process is completing its total execution is called a completion time.
- 3.) Burst Time :- The time required by the process for its execution is called as a Burst time of a process.
- 4.) Turn about Time :- The time difference b/w completion time & arrival time is called as a Turn Around time of a Process.

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

$$\text{TAT} = WT + BT$$

- 5.) Waiting time :- The time difference b/w TAT & burst Time is called as a waiting time of a process.

$$\boxed{WT = TAT - B.T}$$

- 6.) Response Time :- The time difference b/w First Response & Arrival time is called as a Response time of a process.

CPU Scheduling.

CPU SCHEDULING:-

Who : Short term scheduler.

Where : Ready state.

When : 1. Run \rightarrow termination,
Run \rightarrow wait,
Run \rightarrow Ready,

2. Wait \rightarrow Ready.

3. New \rightarrow Ready.

GOAL OF THE CPU SCHEDULING:-

- 1.) To maximize the CPU utilization & throughput of the system.
- 2.) To minimize the Average TAT, Average WT & Average response time of the processes.

1.) FIRST COME FIRST SERVE (FCFS):-

CRITERIA :- Arrival Time.

MODE :- Non Preemptive.

Margin of error - AT Ko carefully Dekhe, if it is continuous or not. (CPU idleness). 25

1.)	P. No	AT	BT		
	1	0	2		
	2	1	3		
	3	2	5		
	4	3	4		
	5	4	6		

P. No	AT	BT	CT	TAT	WT
1	0	2	2	2	0
2	1	3	5	4	1
3	2	5	10	8	3
4	3	4	14	11	7
5	4	6	20	16	10

P ₁	P ₂	P ₃	P ₄	P ₅
0	2	5	10	14 20

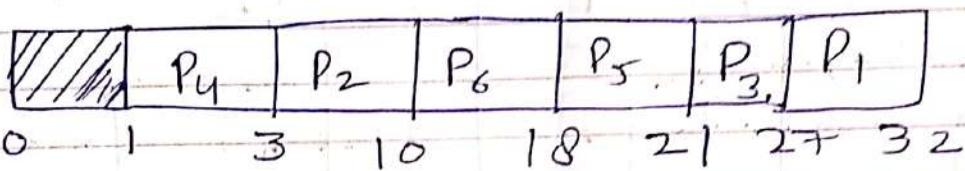
$$\text{Average TAT} = \frac{41}{5} = 8.20$$

$$\text{Average WT} = \frac{21}{5} = 4.20$$

NOTE:- first look at the A.T. carefully.

2.)

P. No	AT	BT	CT	TAT	WT
1	6	5	32	26	21
2	2	7	10	8	1
3	5	6	27	22	16
4	1	2	3	2	0
5	4	3	21	17	14
6	3	8	18	15	7

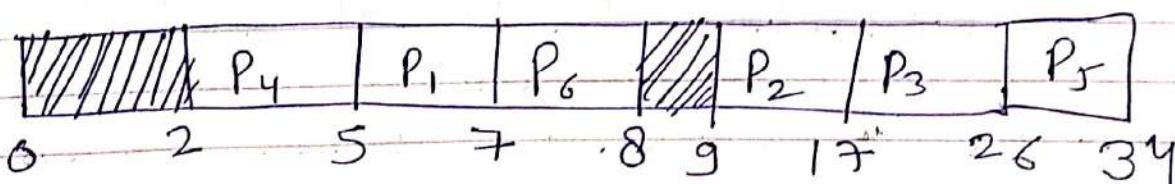


$$\text{Average TAT} = \frac{90}{6} = 15.$$

$$\text{Average WT} = \frac{59}{6} = 9.83$$

3.)

P. No	AT	BT	CT	TAT	WT
1	3	2	7	4	2
2	9	8	17	08	0
3	12	9	26	14	5
4	2	3	5	3	0
5	15	8	34	19	11
6	3	1	8	5	4



$$\text{Average waiting} = \frac{53}{6} = 8.83$$

$$\text{Average WT} = \frac{22}{6} = 3.666$$

Convo effect:- In the FCFS, the first process is CPU bound process & followed by many I/O bound process, then it will have major effect on Average wt of the process. This effect is called Convo effect.

4.)

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

$$\text{Average TAT} = 20$$

$$\text{Average WT} = \frac{38}{3} = 12.67.$$

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

$$\text{Average TAT} = \frac{22}{3} = 7.3$$

2) SHORTEST JOB FIRST (SJF) :-

Criteria :- Burst time.

Mode :- non preemptive.

P No	AT	BT	CT	TAT	WT
1	1	7	8	7	0
2	2	3	14	12	9
3	3	4	18	15	11
4	4	1	9	5	4
5	5	2	11	6	4

	P ₁	P ₄	P ₅	P ₂	P ₃
G	1	8	9	11	14

$$\text{Average TAT} = \frac{45}{5} = 9$$

Pros : gives minimum average response time.

Cons : long running jobs may starve

NOTE :- If the burst time of the process is matching, then schedule the process, which has the lowest arrival time.

(Q1)	P. No	AT	BT	CT	TAT	WT	
	1	24	18	54	36	12	$\frac{312}{6} = 52.83$
	2	15	27	127	112	85	
	3	21	23	160	179	56	
	4	2	19	21	19	0	$\frac{192}{6} = 32$
	5	19	23	77	58	35	
	6	17	15	36	219	4	
					$= \frac{54.66}{6} = 9.11$		$\frac{192}{6} = 32$
					$312/6 = 52$		

P ₄	P ₆	P ₁	P ₅	P ₃	P ₂
0	2	21	36	54	77

SHORTEST REMAINING JOB FIRST

SHORTEST JOB FIRST (S.J.F.)

Criteria :- Burst time

mode :- Preemptive

PNo	AT	BT	CT	TAT	WT
1	1	7	22	21	14
2	2	5	16	14	9
3	3	3	7	4	1
4	4	1	5	1	0
5	5	2	9	4	2
6	6	3	12	6	3
				$= \frac{50}{6} = 8.33$	$= \frac{29}{6} = 4.83$

P ₁	P ₂	P ₃	P ₄	P ₅	P ₃	P ₅	P ₆	P ₂	P ₁
0	1	2	3	4	5	6	7	9	12

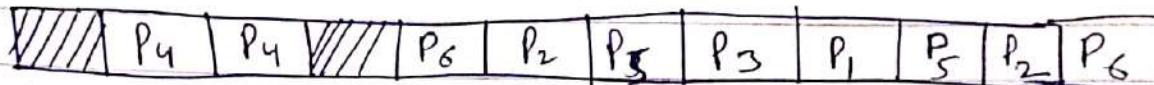
16 22

~~25~~
26

Q.)

PNo	AT	BT	CT	TAT	WT
1	8	2	16	2	0
2	5	7	12	14	7
3	7	1	8	1	0
4	1	2	3	2	0
5	6	4	13	7	3
6	4	9	23	23	14

$$\frac{49}{6} = 8.16 \quad \frac{24}{6} = 4$$

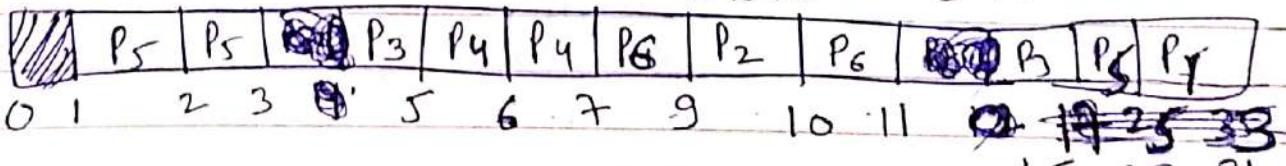


0 1 2 3 4 5 6 7 8 10 13 19 27

(Q)

PNo	AT	BT	CT	TAT	WT
1	11	8	19	12	4
2	9	1	10	1	0
3	3	6	15	12	6
4	5	2	7	2	0
5	1	10	31	30	26
6	7	3	17	10	1

$$\frac{61}{6} = 10.16 \quad \frac{31}{6} = 5.16$$



Use Pencil for Doing calculation work.

NOTE:- Gant chart banaane ke Baad, BT ko Pencil
Mita Do. (updated)

→ In case of SRTF, watch next AT carefully,
(upto which you can run previous process).

~~gate 2016
2 marks~~

P.No	AT	BT	CT	TAT
1	0	10	20	20
2	3	6	10	7
3	7	1	8	1
4	8	3	13	5

$$33/4 = 8.25$$

Solve it by SRTF algorithm?

P ₁	P ₂	P ₃	P ₂	P ₃	P ₁
0	3	7	8	10	13

Q2.) Consider three processes A, B, C to be scheduled as per S.R.T.F. algorithm. The process A is known to be scheduled first, and when 'A' has been running for 7 units of time, the process C has arrived. The process C has run for 1 unit of time, the process B has arrived & completed running in 2 units of time. Then what could be the minimum burst times of the processes A & C?

a) 11 & 4

b) 11 & 3

c) 12 & 3

d) 12 & 4

A remaining time is more than C... time is more than b.

so

$$A \rightarrow 7 + 5 \ 12$$

$$C \rightarrow 1 + \underline{3} = 4$$

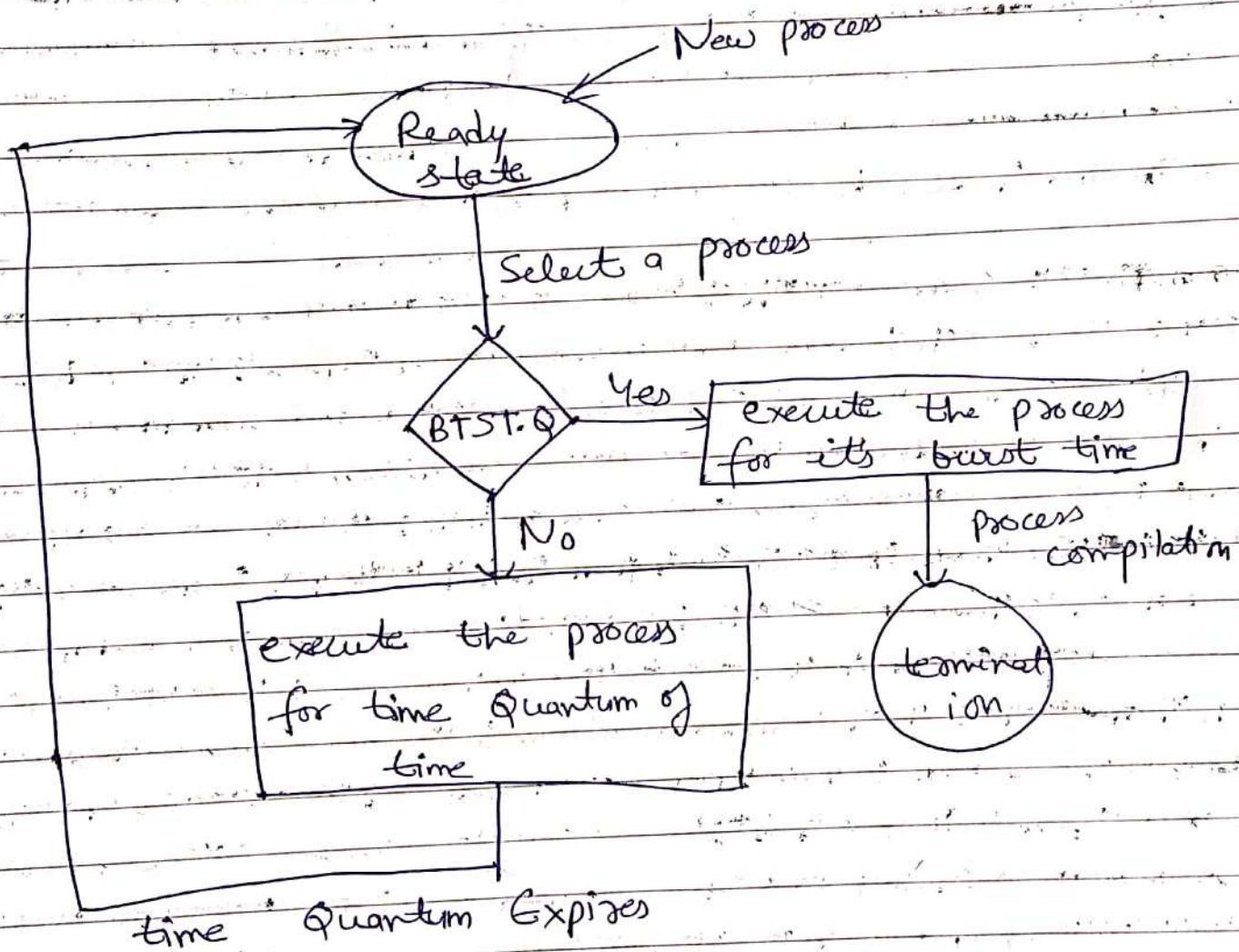
$$B \rightarrow 2 + \cancel{1} - \cancel{1}$$

ROUND ROBIN SCHEDULING:-

(time slice)

Criteria :- Time quantum, Arrival Time

mode :- preemptive



C.S = Context switching

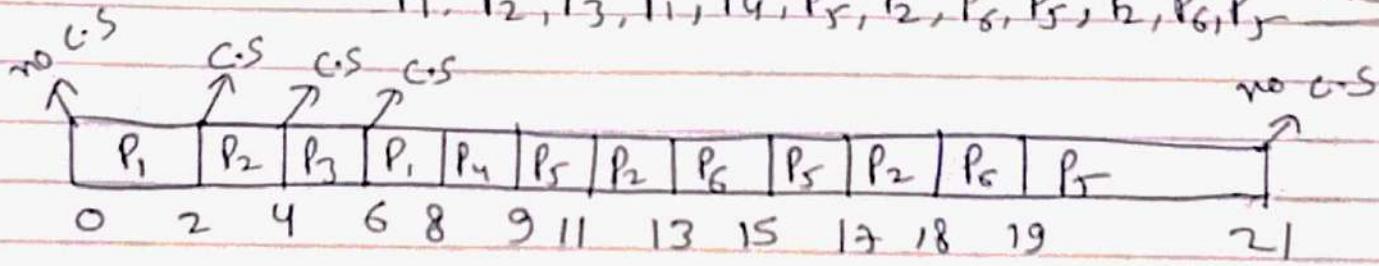
Q1.)

$$T.Q = 2$$

P. No	A.T	B.T	C.T	T.A.T	WT
1	0	4	8	8	4
2	1	5	18	17	12
3	2	2	6	4	2
4	3	1	9	6	5
5	4	6	21	17	11
6	6	3	19	13	10

Ready Queue:-

$$P_1, P_2, P_3, P_1, P_4, P_5, P_2, P_6, P_5, P_2, P_6, P_5 \quad \frac{65}{6} = 10.83 \quad \frac{44}{6} = 7.33$$



$$\text{Average TAT} = 10.83$$

$$\text{Average WT} = 7.33$$

first response - AT

same as first

2.)	R.T AT	P.No.	AT	B.T
	0	1	0	4
	1	2	1	5
	2	3	2	2
	5	4	3	1
	5	5	4	6
	7	6	6	3

$$\frac{20}{6} = 3.33$$

3.)

$$T \cdot Q = 3$$

5 3 3
3 0 8

35

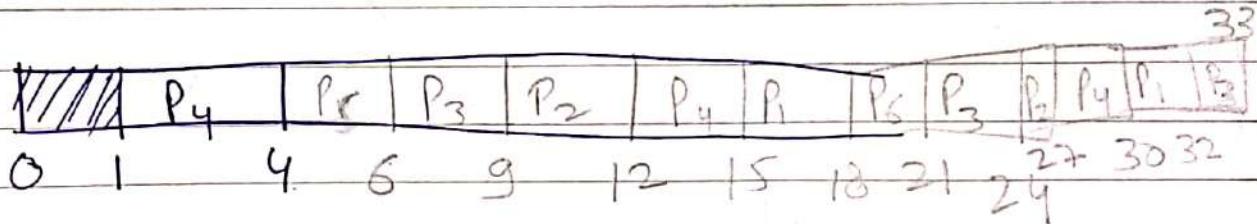
$$\frac{9}{6} = 16 \\ =$$

P.No	AT	B.T	C.T	TAT	WT	RT
1	5	5	32	27	22	10
2	4	6	27	23	17	5
3	3	7	30	20	23	3
4	1	9	30	29	20	0
5	2	2	6	4	2	2
6	6	3	21	15	12	12

$$\frac{120}{6} = 20.33 =$$

Ready Queue:-

$P_4 P_5 P_3 P_2 P_4 P_1 P_5 P_3 P_2 P_4 P_1 P_3$



NOTE:- for calculation of Response Time, check Gantt chart from starting & putting the values in corresponding Process Id.

4.) $T \cdot Q = 2$

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

$$R \cdot Q = P_3 P_6 P_1 P_3 P_4 P_5 P_6 P_2 P_1 P_3 P_2$$

P ₃	P ₆	P ₁	P ₃	P ₄	P ₅	P ₆	P ₂	P ₁	P ₃	P ₂
0	2	4	6	8	9	11	13	15	16	18

A

NOTE :- (i) If the round robin, if the time quantum is less, no. of context switching will increase and No. of the response time will decrease.

(ii) If the time quantum is large, then no. of context switches will decrease & response time will be more.

(iii) If the time quantum is greater than all the burst time, the round robin similarly behaves like a FCFS algorithm.

Q.) Consider a system, which has 4 processes, P_1, P_2, P_3, P_4 all arriving in the ready queue in the same order at time 0, The Burst time requirements of these processes are:-

B.T - 4, 1, 8, 1 respectively.

Then what is the completion time of Process P_1 , assuming round scheduling with $T.Q = 1$?

P. No	AT	BT	CT	TAT	WT	RT
1	0	4	9			
2	0	1	2			
3	0	8	13			
4	0	1	4			

Ready Queue :-

$P_1 P_2 P_3 P_4 P_1 P_3 P_1$

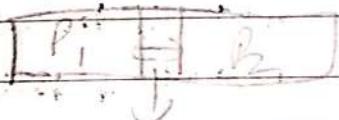
P ₁	P ₂	P ₃	P ₄	P ₁	P ₃	P ₁	P ₁ + P ₃	P ₁
0	1	2	3	4	5	6	7	8

~~Set 1998~~

Q.) Consider n processes sharing the CPU in round robin algorithm. The context switching time is " s " units, then what must be the time quantum "q", such that the no. of context switches are reduced, but at the same time each process is guaranteed to get its turn/job at the CPU after every " t " seconds of time?

Assume AT = 0.

a.) $q = \frac{t - ns}{n+1}$

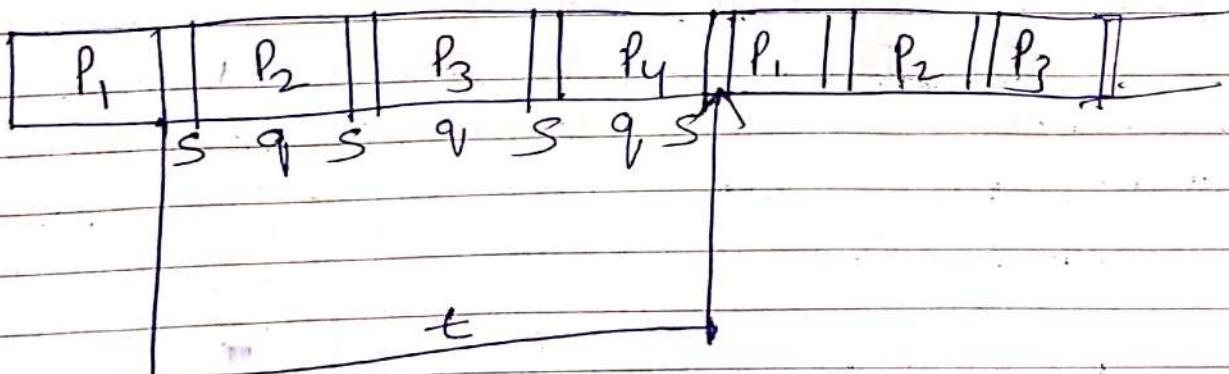


b.) $q = \frac{t + ns}{n-1}$

Stimes

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

d.) $q = \frac{t + ns}{n+1}$



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

$$(n-1) \times q = t - ns$$

LONGEST JOB FIRST (L.J.F):-

Criteria :- Burst time.

Mode :- ^{Non} preemptive.

$$41/5 = 8.2$$

P.T	A.T	B.T	CT	TAT	WT
1	0	2	2	2	0
2	1	6	23	22	16
3	2	7	9	7	0
4	3	4	27	24	20
5	4	8	17	13	5
				68	41

$$\frac{68}{8} = 13.60$$

P ₁	P ₃	P ₅	P ₂	P ₄
0	2	9	17	23

$$\text{Average TAT} = 13.60$$

$$\text{Average WT} = 8.20$$

NOTE:- If the burst time of the processes are matching, schedule the process which have lowest arrival time.

Q1.) P. No

AT

BT

CT

TAT

WT

1

24

19

89

65

46

2

17

23

70

53

30

3

21

16

121

109

84

4

2

18

20

18

0

5

19

16

105

86

20

6

13

27

47

34

27

P₄P₆P₂P₁P₅P₃

0 2

28

47

70

89

105

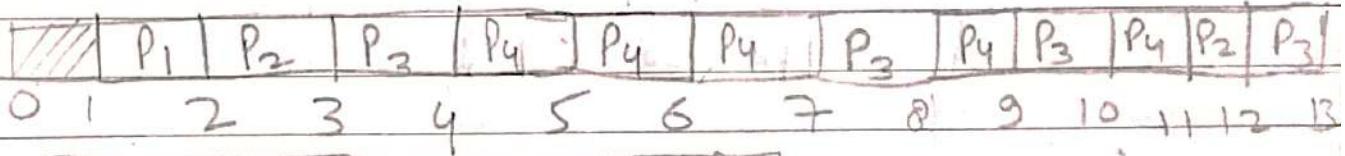
121

6.) LONGEST REMAINING TIME FIRST (L.R.T.F.)

CRITERIA:- Burst time.

MODE :- preemptive.

PNo	AT	BT	CT	TAT	WT
1	1	2	18		
2	2	4	19	1	
3	3	6	20	1	
4	4	8	21	1	



$$\text{Average TAT} = \frac{14 + 15 + 16 + 17}{4} = 15$$

$$\text{Average WT} = 10$$

GATE

12

23

25

23

P.No	AT	BT	CT	TAT	WT
2 2	2 1	0 2	12 11	12 11	10
4 3	2 2	0 4	13 13	13 13	9
4 4	2 3	0 8	14 14	14 14	16

39/7

35/35

what is average TAT using LRTF
algo?

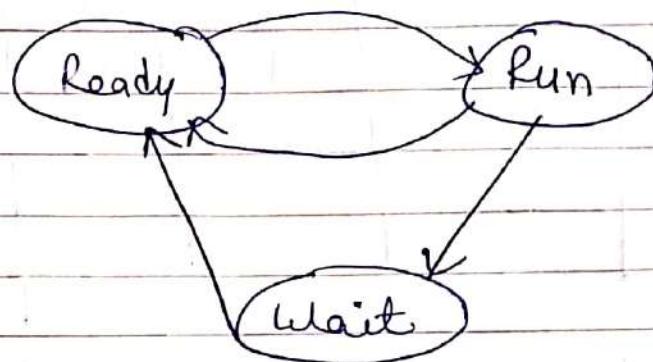
11.66

P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇	P ₈	P ₉	P ₁₀	P ₁₁	P ₁₂	P ₁₃	P ₁₄	P ₁₅	P ₁₆	P ₁₇	P ₁₈	P ₁₉	P ₂₀
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

$$TAT = \frac{13}{7}$$

$$WT = \frac{25}{7} = 3.57$$

CPU TIME, I/O Time Concept:-



Life cycle:-

CPU time

I/O time

CPU time

I/O time

|

|

|

Complete

Cases:-

- 1.) CPU ✓
- 2.) I/O ✓
- 3.) CPU, I/O ✓
- 4.) CPU, I/O, I/O ✓
- 5.) I/O, I/O, I/O ✓
- 6.) CPU, CPU, I/O ✗

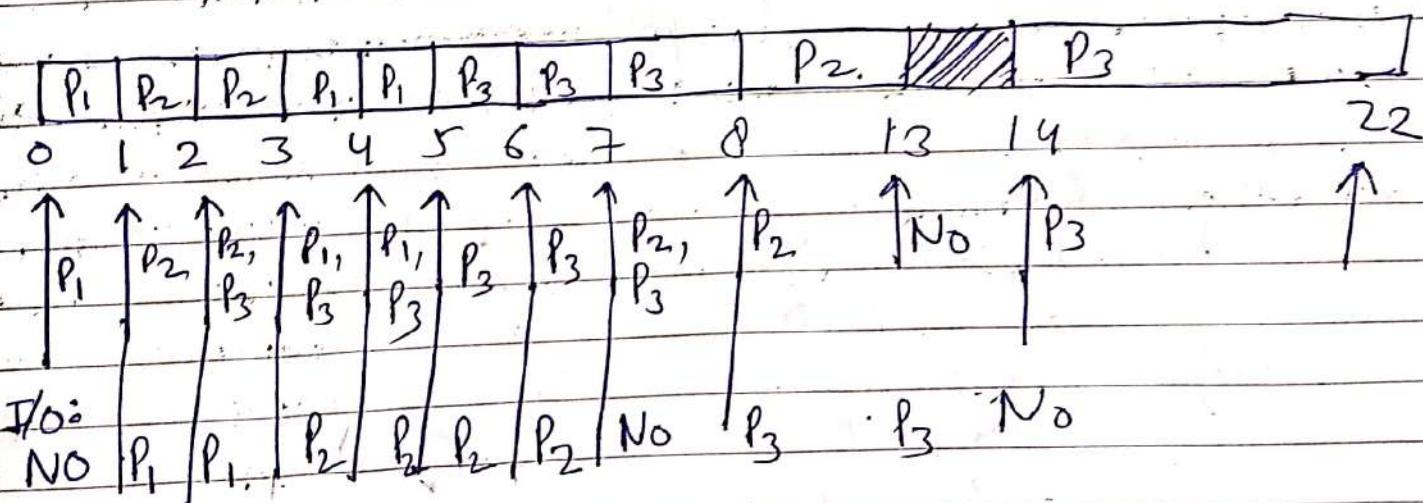
Q.)

		Execution Time				
A.T.	P. No.	CPU Time	I/O time	C/I time	C.T	
0	1	10	2	2	5	
1	2	210	4	5	13	
2	3	32X0	6	8	22	

What is the completion time of processes P_1, P_2, P_3 using S.R.F.F algorithm?

NOTE:- 1.) The process first spends CPU time followed by I/O time and followed by CPU time again.

2.) I/O of the processes can be overlapped as much as possible.



Execution Time

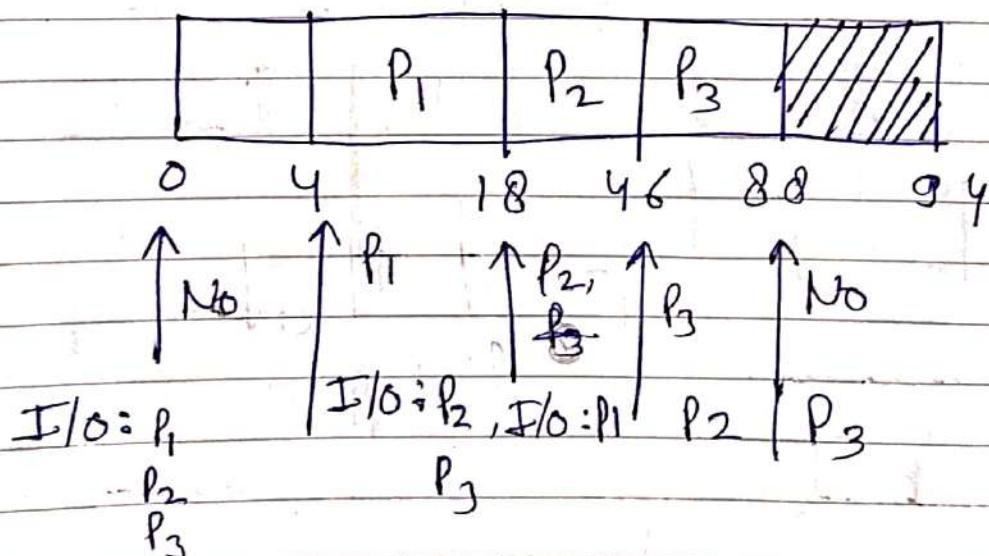
Q2.)

P.No	AT	I/O	CPU Time	I/O Time	Completion Time
		CPU Time			
1	0	40	140	240	20
2	0	840	28	4	50
3	0	1280	42	6	94

What is the completion time using SRTF?

NOTE:- 1) The process first spends I/O time followed by CPU time and followed by I/O time again.

2) I/O of the processes can be overlapped as much as possible.



(Q1) Consider three processes, all arriving at the time '0', with the total execution time of 10, 20 and 30 units respectively. Each process spends first 20% of execution time doing I/O, the next 70% of execution time doing computation, and last 10% of time doing I/O again. The O.S. uses S.R.T.F algorithm. All I/O operations are overlapped as much as possible. For what percentage of the does the CPU remain idle?

- a) 0 %.
- b) 10.6 %.
- c) 30.0 %.
- d) 89.4 %.

Execution Time					
P.No	A.T	I/O	CPU	I/O	Completion time
1	0	2.0	7	1.0	10
2	0	4.0	14	2	20
3	0	6.0	21	3	30

47

S.R.T.F

2 seconds

3 seconds

(5)



Q. 2. 9 23 44

45

No | P₁ | P₂ | P₃ | No

I/O ↓ I/O ↓ I/O ↓ I/O ↓ I/O ↓ P₃

P₁ P₂, P₂ P₃, P₃

$$= \frac{5}{47} \times 100$$

47

Q2.)

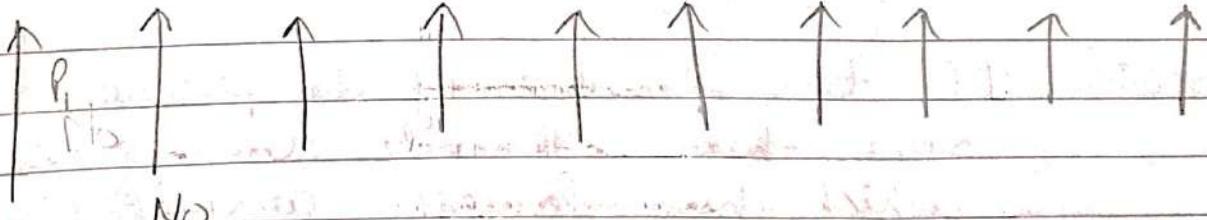
P.No	AT	CPU Time	I/O Time	CPU Time	C.T
1	0	5	2	5	12
2	3	2	2	2	31
3	7	8	0	0	17
4	25	9	2	1	39

What is the completion time of P_1, P_2, P_3, P_4 using S.R.T.F algorithm?

NOTE :- 1) The process first spends CPU time followed by I/O Time followed by CPU time

2) I/O of the processes can be overlapped as much as possible

P ₁	P ₁	P ₂	P ₃	P ₁	P ₃	P ₄	P ₂
0	3	15	7	10	12	17	25



I/O No

No

28
24
32

PRIORITY BASED SCHEDULING:-

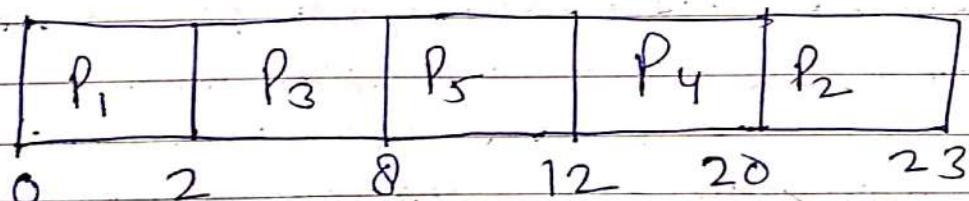
CRITERIA:- PRIORITY

MODE:- NON PREEMPTIVE

PRIORITY	P.No	AT	B.T	TAT	WT	C.T
2	1	0	2	2	0	2
4	2	1	3	22	19	23
6	3	2	6	6	0	8
5	4	3	8	17	9	20
high \leftarrow 7	5	4	4	8	4	12

$$\frac{55}{5} = 11 \quad \frac{32}{5} = 6.4$$

Assuming :- Highest Number have \Rightarrow highest priority.



$$\text{Average TAT} = 11$$

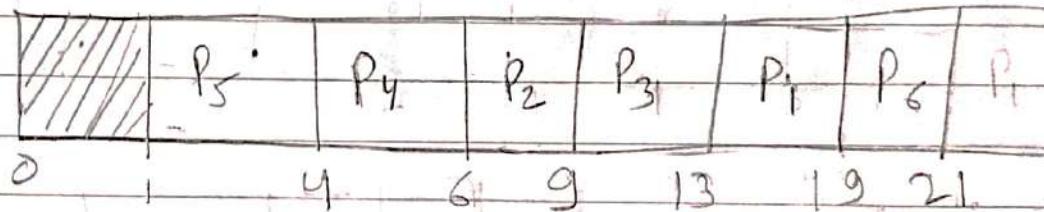
$$\text{Average WT} = 6.4$$

NOTE:- If the priorities of the processes are same, then schedule the process which has lowest arrival time.

Q.)

Priority	P.No	AT	BT	CT	TAT	WT
4	1	4-	6	10	5	9
7	2	6	3	9	3	0
6	3	3-	4	13	10	6
6	4	2-	2	6	4	2
1	5	1	3	4	3	0
3	6	2-	2	21	19	17

$$\frac{54}{6} = 9 \quad \frac{34}{6} = 5.66$$



Average TAT = 9

Average WT = 5.66

Q1.)

Criteria : priority .

mod : preemptive .

Priority	P. No	A.T	B.T	C.T	T.D.T
3	1	0	4	19	19
2	2	1	6	25	24
4	3	2	5	17	15
6	4	3	3	13	10
8	5	4	7	11	7

$$= 15 \quad 25$$

P.	P ₁	P ₃	P ₄	P ₂	P ₄	P ₃	P ₁	P ₂
0	1	2	3	4	11	13	17	19

1 2 3 4 11 13 17 19 25

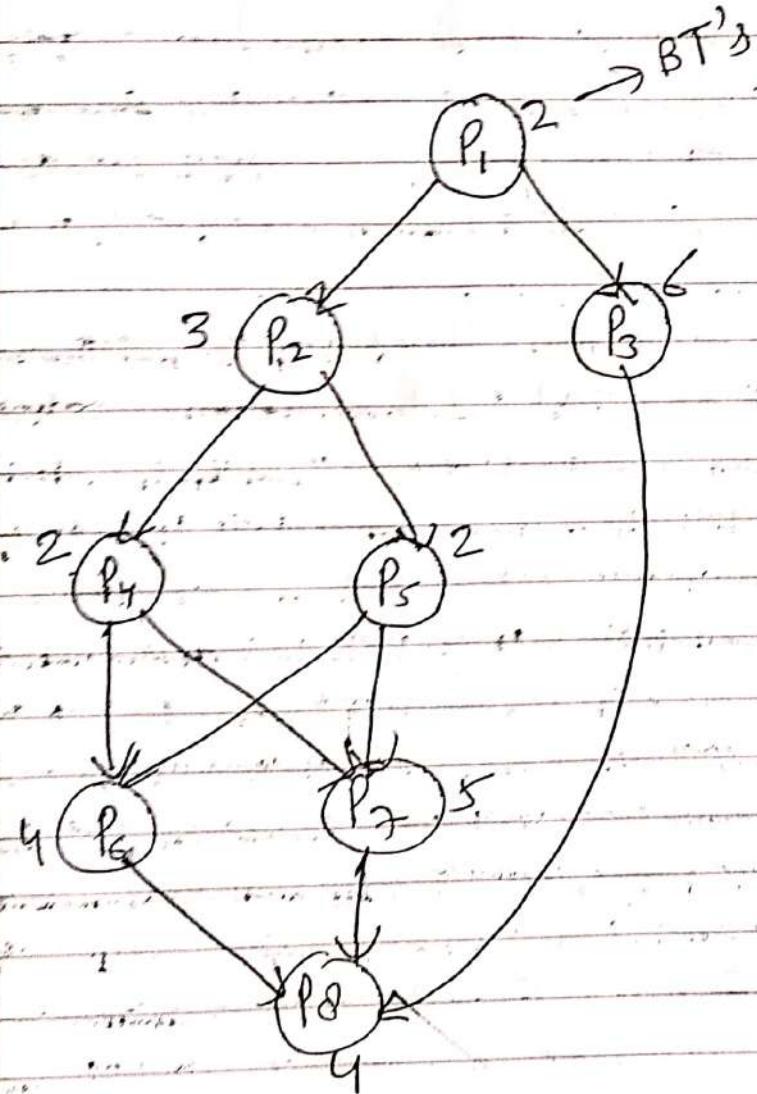
25.66 18.16

Q2.)

	Priority	P. No	AT	BT	CT	TAT	WT
	2 -	1	1	9			
19	3 -	2	2	8			
	5	3	3	7			
	2 -	4	0	6			
	6	5	4	8			
High	→ 7	6	5	7			

P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇	P ₈	P ₉	P ₁₀	P ₁₁	P ₁₂
0	1	2	3	4	5, 12	13	20	26	33	39	4

a.) Consider the following dependency graph between the processes:-



At what time, all the processes complete their execution by using 2 CPU

processors?

- a.) 17
- b.) 18
- c.) 19
- d.) 20
- e.) 21.

NOTE :- 1.) One process can not use '2 CPUs' at the same time.

CPU ₁	P ₁	P ₂	P ₄	P ₅	P ₅	P ₆	/	/	/
CPU ₂	/	/	P ₃	P ₃	P ₃	/	P ₇	P ₇	P ₈
	0	2	5	7	8	9	13	14	18

HIGHEST RESPONSE RATIO NEXT (HRRN):

CRTIERIA:- Response Ratio

MODE:- Non preemptive

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

W = Waiting Time

S = Service Time or Burst time.

NOTE:- The above algorithm favours the shortest jobs and limits the waiting time of longer jobs.

Eg:-

P.No	AT	BT	CT	TAT	WT
1	0	3	3	3	0
2	2	6	9	7	1
3	4	4	13	9	5
4	6	5	20	14	9
5	8	2	15	7	5

$$\frac{40}{5} = 8$$

$$\frac{20}{5} = 4.$$

P ₁	P ₂	P ₃	P ₄	P ₅
----------------	----------------	----------------	----------------	----------------

0 3 9 13 15 20

13

$$RP_3 = \frac{W+T}{S} = \frac{5+4}{4} = \frac{9}{4} = 2.25$$

at time 9

$$RP_4 = \frac{W+T}{S} = \frac{3+5}{5} = \frac{8}{5} = 1.6$$

$$RP_5 = \frac{W+T}{S} = \frac{1+2}{2} = \frac{3}{2} = 1.5$$

at time 13

$RP_4 = \frac{7+5}{2} = \frac{12}{5} = 2.4$	P_4
$RP_5 = \frac{15+2}{2} = \frac{17}{2} = 8.5$	P_5

$$\text{Average TAT} = \frac{40}{5} = 8$$

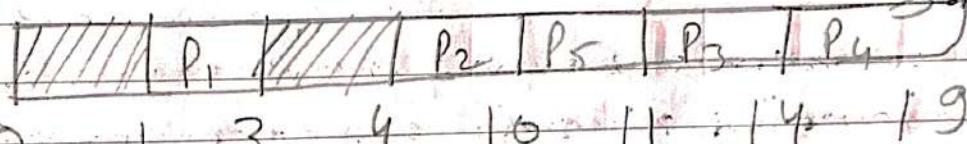
$$\text{Average WT} = \frac{20}{5} = 4.$$

Eg:

P.NO	AT	BT	CT	TAT	WT
1	1	2	3	2	0
2	4	6	10	6	0
3	5	3	14	9	6
4	6	5	19	43	8
5	8	1	11	3	2
					16

$$33 = 6.60$$

$$= 3.2$$



$$RP_3 = \frac{5+3}{3} = \frac{8}{3} = 2$$

$$\text{at time } 10 \quad RP_4 = \frac{4+5}{5} = \frac{9}{5} = 2$$

$$RP_5 = \frac{2+1}{3} = \frac{3}{3} = 1$$

$$\text{at time } 11 \quad RP_3 = \frac{6+3}{3} = \frac{9}{3} = 3$$

$$RP_4 = \frac{5+5}{5} = \frac{10}{5} = 2$$

$$\text{Average TAT} = 6.60$$

$$WT = 3.2$$

P-85, Q-29

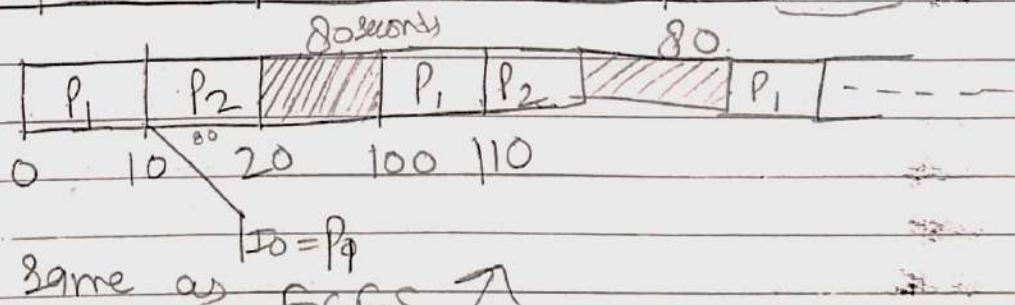
GATE

Consider a system has only two processes, both of which alternate 10ms of CPU burst with 90ms of I/O burst. Both the processes created nearly at the same time. The I/O of both the processes can proceed in parallel which of the following scheduling strategies will result in least CPU Utilization over a long period of time?

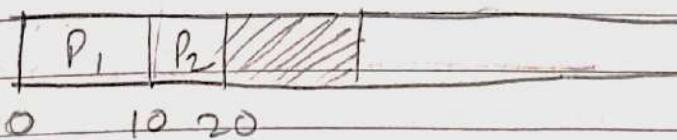
- a.) F.C.F.S.
- b.) S.R.T.F.
- c.) Priority
- d.) Round Robin with time quantum of '5'ms.

R.No	CPU	I/O	CPU	I/O	CPU
P ₁	10	90	0	90	0
P ₂	10	90	0	90	0

a.) F.C.F.S :-

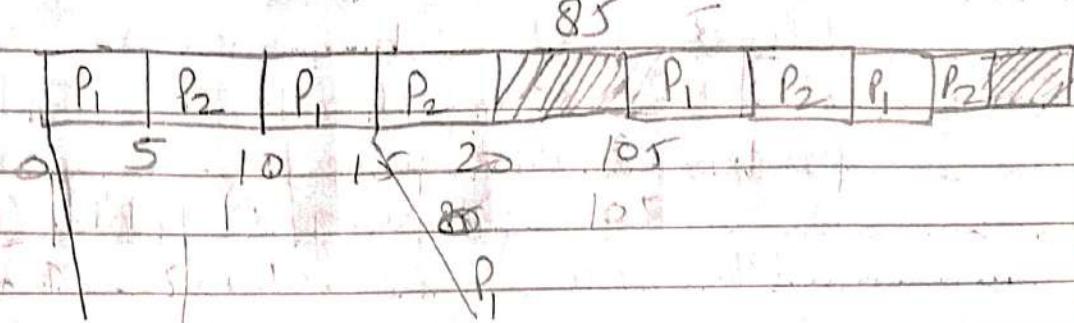


b.) S.R.T.F



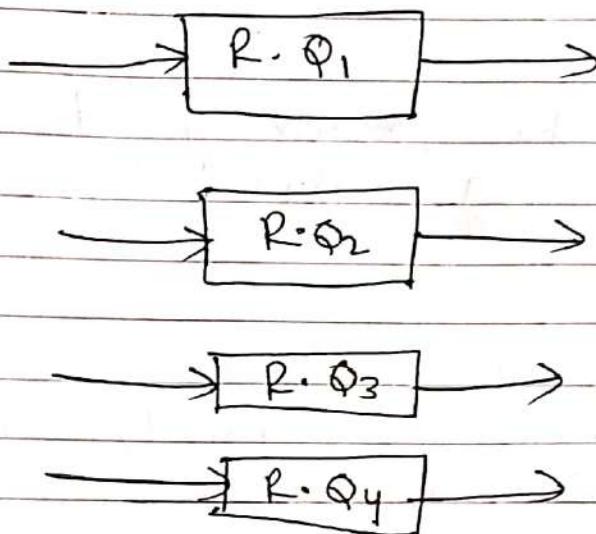
c.) Priority :- Same as S.R.T.F.

d) Round Robin with time quantum of 5 ms.



Answer is option d as the CPU idle time in Round Robin is 85 which in other scheduling algorithm have 80, 80. Round Robin will result in low CPU Utilization.

MULTI LEVEL X QUEUE X SCHEDULING:-



→ Depending on the priority of a process, in which particular ready queue, the process has to be placed will be decided.

→ The high priority processes will be placed in top priority ready queue & the low priority processes will placed on a low priority ready queue.

→ only on the completion of process from the top level ready queue, the further level ready queue process will be scheduled.

→ If this is strategy is followed then the processes which are placed in the bottom level ready queue will suffer from starvation.

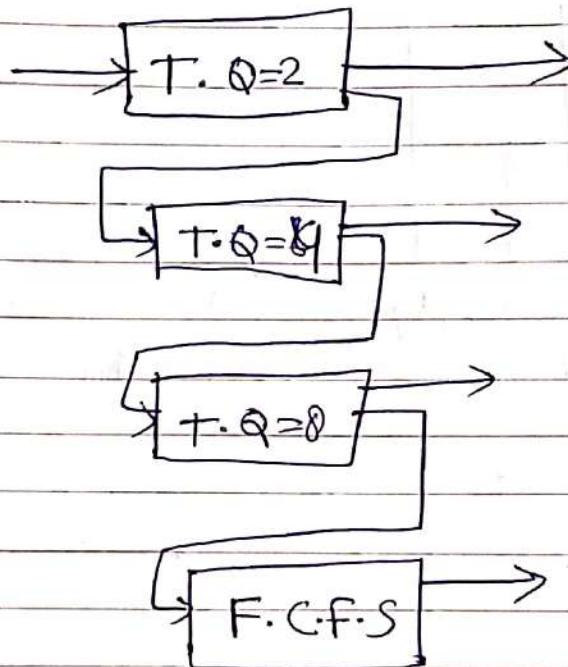
STARVATION: The Indefinite waiting of a process is called as a Starvation.

NOTE:- To avoid the problem of starvation, the concept of Ageing will be used.

Ageing:- If the waiting time of a process increases then the priority of a process will be increased.

If the age of a process increases by increasing the priority, the process will definitely get a chance to execute and starvation problem will be avoided.

MULTI LEVEL FEED BACK QUEUE SCHEDULING



NOTE:- In the above algorithm, every process will definitely get a chance to execute but still there is a possibility for the process to go into starvation.

P.No	AT	B.T
1	0	2
2	1	3
3	2	5
4	3	6
5	4	8
6	5	7

what is the throughput of the system using FCFS algorithm?

P ₁	P ₂	P ₃	P ₄	P ₅	P ₆
0	2	5	10	16	24

$$\text{throughput} = \frac{\text{No. of processes}}{\max(\text{C.T}) - \min(\text{A.T})}$$

$$= \frac{6}{31 - 0} = \frac{6}{31} = 0.19.$$

$$= 19\%$$

$$\text{Schedule length} = \boxed{\max(\text{C.T}) - \min(\text{A.T})}$$

P.No	A.T	B.T
1	3	2
2	9	5
3	2	3
4	12	6
5	13	8
6	3	1

What is the throughput of the system using F.C.F.S algorithm?

	P ₃	P ₄	P ₆	P ₂	P ₄	P ₅
0 2 5 7 8 9 14 20 28						

$$\frac{0.23}{26} \times 100 = 0.88\%$$

$$= \frac{6}{28-3} = \frac{6}{25} - [0.23] \\ = 23\%.$$

NP - Non preemptive

Algorithm	Starvation
1.) FCFS	No.
2.) NP \rightarrow SJF	Yes.
3.) SRTF	Yes.
4.) * RR *	No.
5.) NP \rightarrow LTF	Yes.
6.) LPTF	No.
7.) NP \rightarrow Priority	Yes
8.) Pre- LPTF Priority	Yes
9.) H.R.R.N	No.
10.) M.L.Q	Yes
11.) M.L.F.B.Q	Yes

NOTE: Majority of the operating systems
practically implement Round Robin
scheduling.

22/10/11

65

NOTE: In the shortest job first scheduling Algorithm, the burst times of the processes are expected by using the following formula

$$T_{n+1} = \alpha \cdot T_n + (1-\alpha) \cdot \bar{T}_n$$

T_{n+1} \rightarrow next expected burst time

T_n \rightarrow previous expected burst time.

\bar{T}_n \rightarrow previous actual burst time

' α ' is a parameter which controls, the relative weight of recent and past history.

$$0 \leq \alpha \leq 1$$

(32)

Process	Burst Time
P ₁	5
P ₂	8
P ₃	3
P ₄	5

(33)

$$\alpha = 1$$

$$T_S = \alpha \cdot T_y + (1-\alpha) T_y$$

$$T_S = 0.5 + 0$$

$$T_y = 10$$

$$\alpha = 0.5$$

$$T_2 = 0.5 \times 5 + 0.5 \times 10$$

$$= 2.5 + 5$$

$$= 7.5$$

$$T_3 = 0.5 \times 8 + 0.5 \times 7.5$$

$$= 4 + 3.75 = 7.75$$

$$T_4 = 0.5 \times 3 + 0.5 \times 7.75$$

$$= 1.5 + 3.875 = 5.375$$

$$T_5 = 0.5 \times 5 + 0.5 \times 5.375$$

$$= 2.5 + 2.6875$$

$$= 5.1875$$

(a)

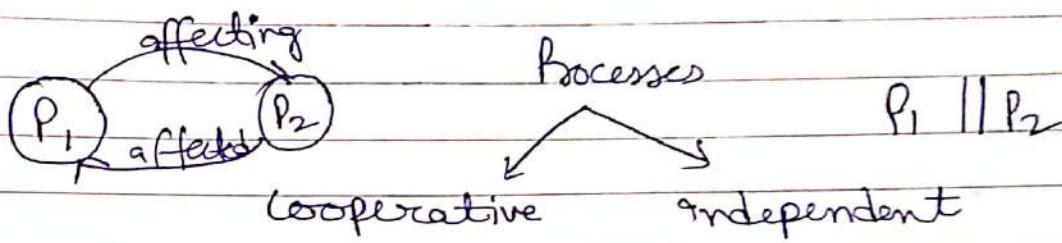
~~2.6875~~
~~2.6875~~
~~5.1875~~

(b) 5

M.O.S by
D.S. Temarainen

SYNCHRONIZATION:

Processes are categorized into two types



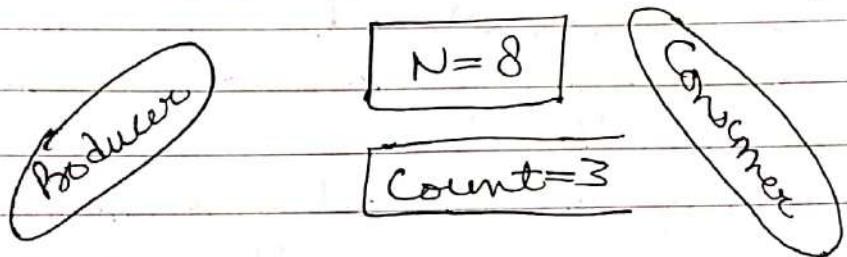
The execution of one process affects or affected by other process, then those processes are said to be cooperative process, otherwise they are said to be independant process.

Understanding x Synchronization:-

- 1.) The problems arises NOT having Synchronization between the processes.
- 2.) the conditions to be followed by to achieve synchronization.
- 3.) The solutions (wrong and Right).

→ Problem Arriving, Purging, Synchronization

The PRODUCER - CONSUMER :-



in	3	0	
	2	4	
	3	2	
	4	A	
	5		
	6		
	7		

Buffer [0...N-1]

Analysis :-

Item P = 'A'
Item C = 'X'

P → I

P → II

R_P [3, 4]

C → I

C → II

C → III

R_C [3, 2]

```

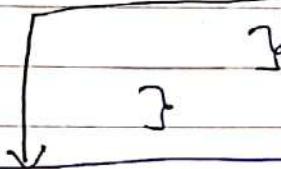
int count = 0;
void producer(void)
{
    int item;
    while(true)
    {
        produce-item(item);
    }
}

```

while (count == N);

Buffer[in] = Item;
 $in = (in + 1) \bmod N;$

Count = Count + 1;



I loads Rp, m[Count]

II. INC R_P

III. Store m[Count], Rp

void consumer(void)

{

int itemc;
 while (true)

{

I. Load R_C, m[Count]

II. DEC R_C

III. Store m[Count], R_C

while (Count == 0);

Itemc = Buffer[out];

out = (out + 1) $\bmod N;$

Count = Count - 1;

process-item(itemc);

}

- IN is a variable, used by the producer to identify the next empty slot in the buffer.
- OUT is a variable, used by the consumer to identify, where it has to consume the item.
- COUNT is a variable used by both producer and consumer to identify the number of items present in the buffer at any point of time.

Shared Resources:

1. Buffer
2. count variable.

Two conditions to be followed:-

- 1.) When the Buffer is full, the producer is not allowed to produce the item in buffer.
- 2.) When the Buffer is empty, the consumer is not allowed to consume the item from buffer.

UNIVERSAL ASSUMPTION :-

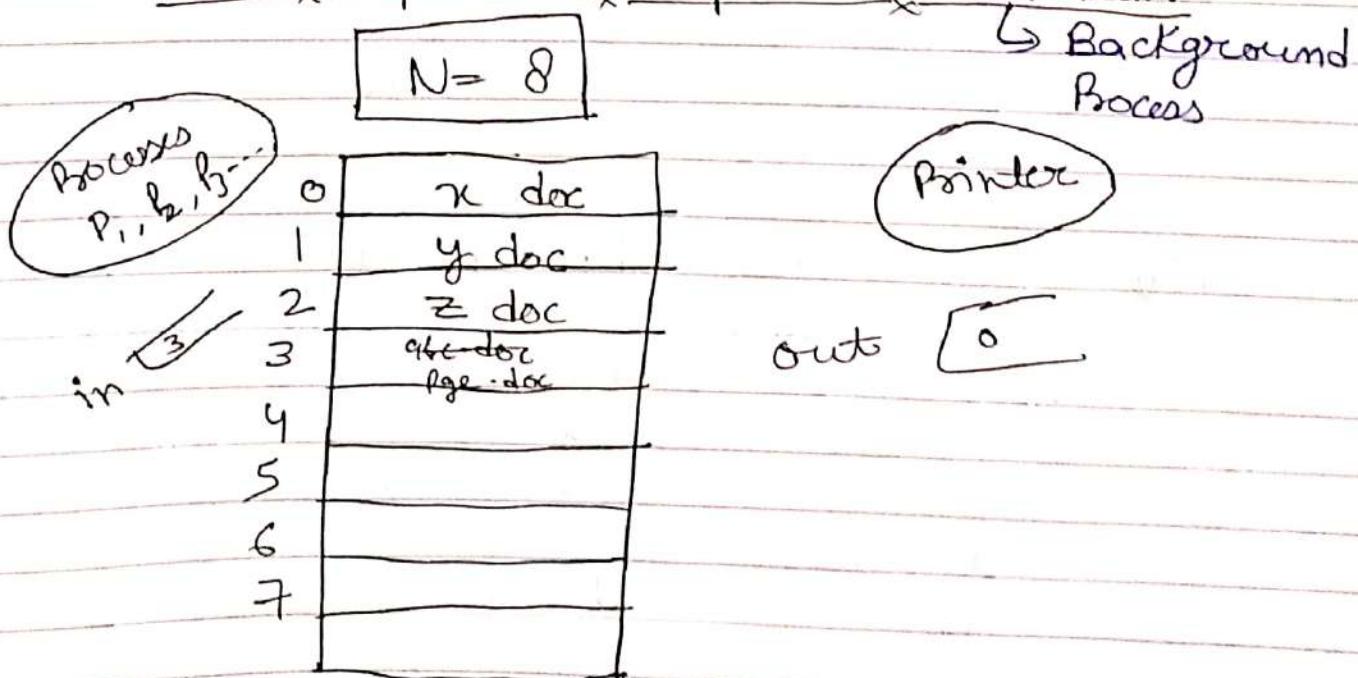
The running process can get preempted at any point of time after completion of current instruction.

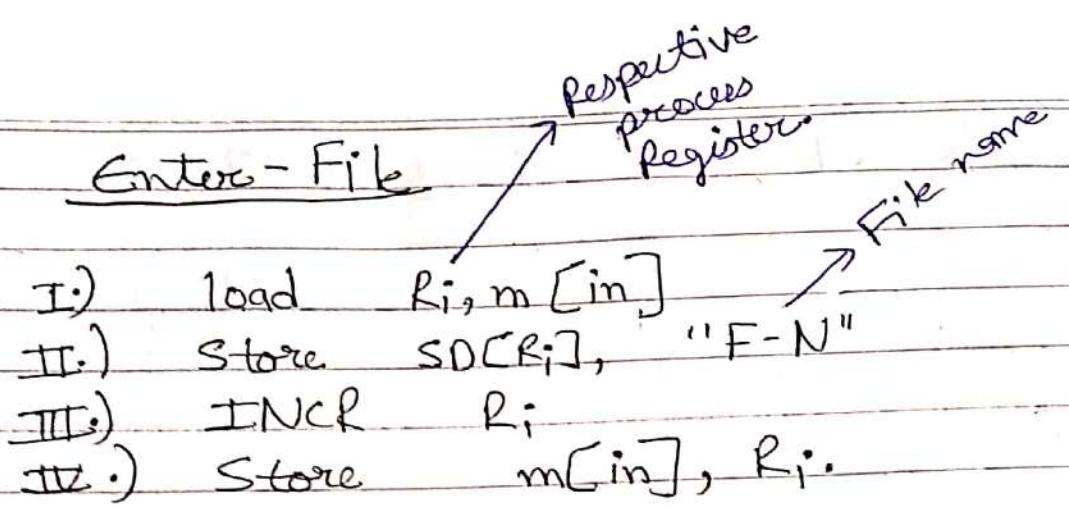
Analysis :-

Problem

- 1) Inconsistency :- The producer and consumer are not properly synchronized while sharing the common variable, COUNT, hence it is leading to the problem of inconsistency.

The pointer spooler Daemon :-





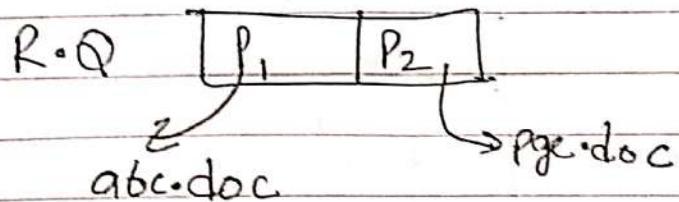
NOTE:- "In" is a variable used by all the processes to identify the next empty slot in the spooler directory.

"out" is a variable used by the printer to identify from where it has to print the document.

Shared resources :-

- 1.) "in" variable.
- 2.) Spooler directory.

Analysis:-



P → I

P → II

P → III

— P₂ → I

P₂ → II

R ₁	3, 4
----------------	------

R ₂	1, 3
----------------	------

1.) Inconsistency :-

7-3

2.) Loss of data :- The processes are not properly synchronized while sharing the common variable "IN", hence it is leading to the Problem of loss of data.

3.) Deadlock : If the processes are not properly synchronized while sharing common variable or common resources then also possible for Deadlock.

Definition :-

1.) CRITICAL SECTION :- The portion of program text, where shared variables or shared resources will be placed.

Eg :- $\text{count} = \text{count} + 1;$
 $\text{count} = \text{count} - 1;$

2.) NON CRITICAL SECTION :- the portion of program text, where the independent code of process will be placed.

Eg :- $\text{in} = (\text{in} + 1) \bmod N;$
 $\text{out} =$

3.) RACE CONDITION :- the final value of any variable depends on execution sequence of the processes. These type of condition is called as race condition.

count = 4

count = 2

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

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

OK

NOTE :- To avoid the problem of race condition, only one process should be allowed to enter critical section at any point of time.

Conditions to be followed to achieve synchronization :-

1.) Mutual Exclusion :-

→ No two process may be simultaneously present inside the critical section at any point of time.

→ Only one process is allowed to enter into critical section at any point of time.

2.) Progress :- → No process running outside the critical

section should lock the other interested process from entering into critical section when

- If there is only one process trying to enter into Critical section, then it should be definitely allowed to enter into Critical section.
- If two or more processes are trying to enter the critical section, then one process should be definitely allowed to enter into Critical section.

3.) Bounded waiting:

- No process should have to wait forever to enter into critical section.
- There should be a bound on getting chance to enter into critical section.
- Some process is indefinitely waiting to enter into critical section, because critical section is always busy by some other processes, this situation should not arise.
- If the bounded waiting is not satisfied, then it is possible for starvation.

4.) No assumptions related to hardware & the processor speed.

Solutions:-

I.) Software Type:-

- a.) Lock variables
- b.) strict alteration of Deckers algo
- c.) petersons algorithms.

II.) Hardware Type:-

- a.) TSL instruction set
 - ↳ test and set lock

III.) O.S. Type:-

- a.) Counting semaphore.
- b.) Binary semaphore.

IV.) Programming language compiler support
type:-

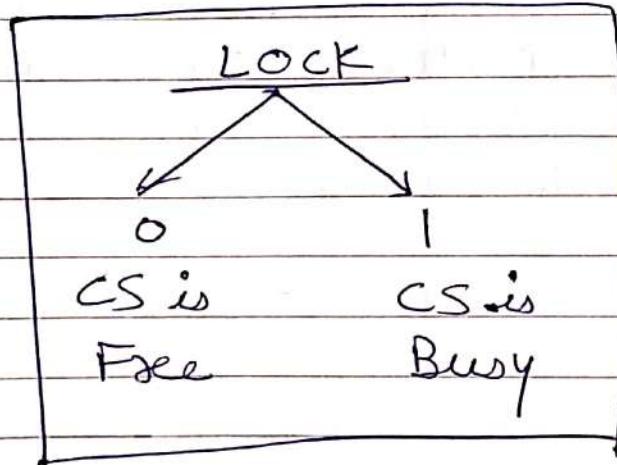
- a.) monitors.

Solution

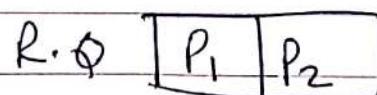
I) LOCK X VARIABLES :-

Entry Section :- *respective process register*

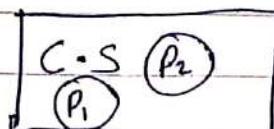
- I. Load $R_f, m[lock]$
- II. CMP $R_f, \#0$
- III. JNZ to step I
- IV. store $m[lock], \#1$
- V. C.S.
- VI. Store $M[lock], \#0$



Analysis :-



$$Lock = 0, 1, 1$$



$$\begin{aligned} P_1 &\rightarrow I \\ P_1 &\rightarrow II \\ P_1 &\rightarrow III \end{aligned}$$



$$\begin{aligned} P_2 &\rightarrow I \\ P_2 &\rightarrow II \end{aligned}$$



$P_2 \rightarrow IV$

$P_2 \rightarrow V$

$P_1 \rightarrow IV$

$P_1 \rightarrow V$

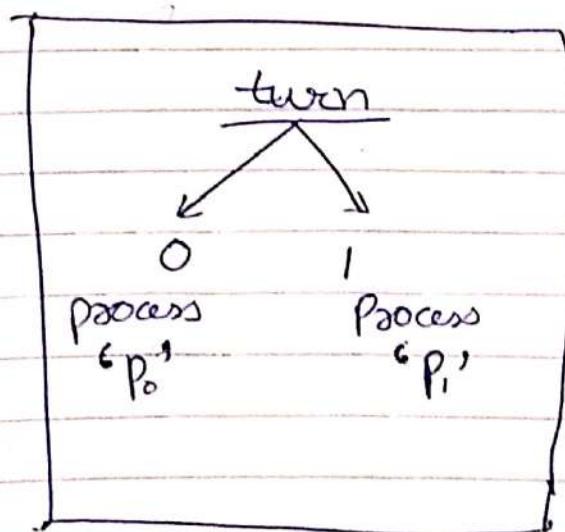
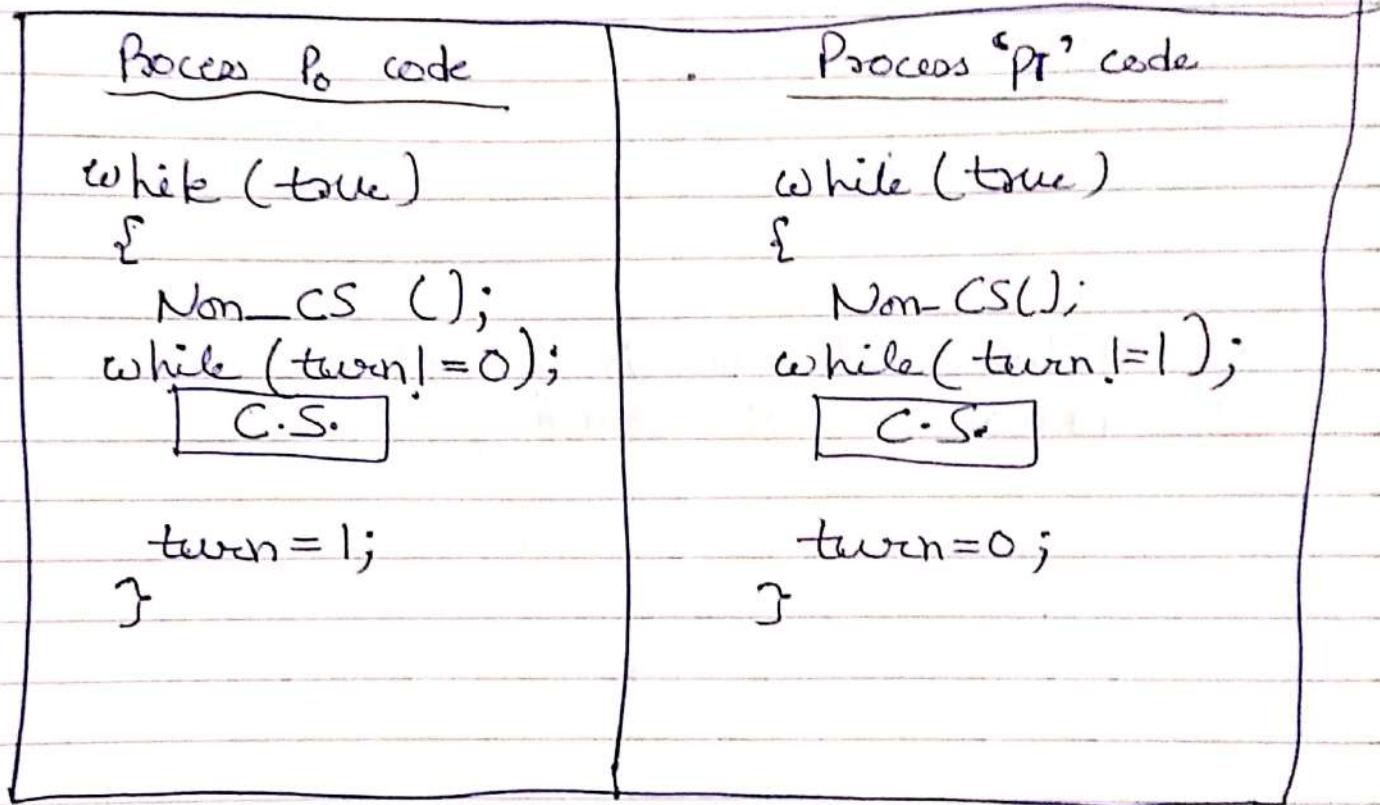
Mutual exclusion is not satisfied

NOTE:- We have proved that both the processes are enter into C.S. at the same time and Mutual exclusion is not satisfying, Hence solution is considered to be incorrect.

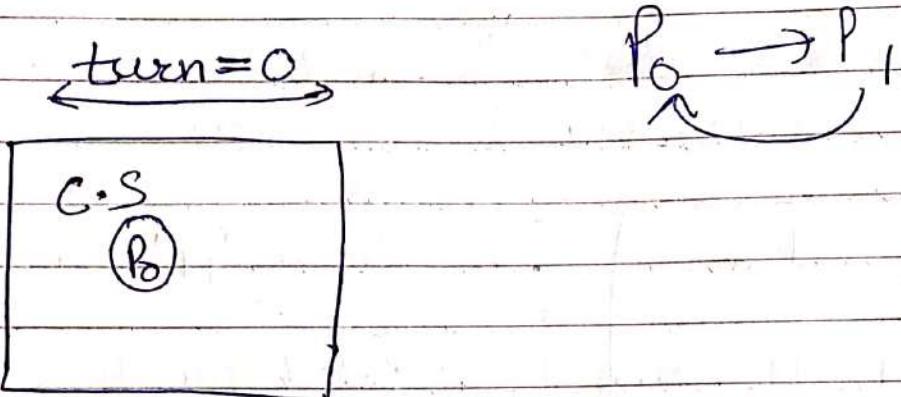
(b)

STRICT ALTERATION OR DECKERS ALGORITHM:

(process takes "turn" to enter into C.S.)



Analysis :-



- (-1-) Mutual exclusion is satisfied.
- (-2-) Progress is not satisfy.

IMPORTANT POINTS:-

- 1.) The preemption is just a temporary stop. The process will come back and continue the remaining execution.
- 2.) If there is any possibility of solution becoming wrong by taking a preemption then consider the preemption.
- 3.) If any solution have deadlock, then the progress is not satisfied.

Page-94

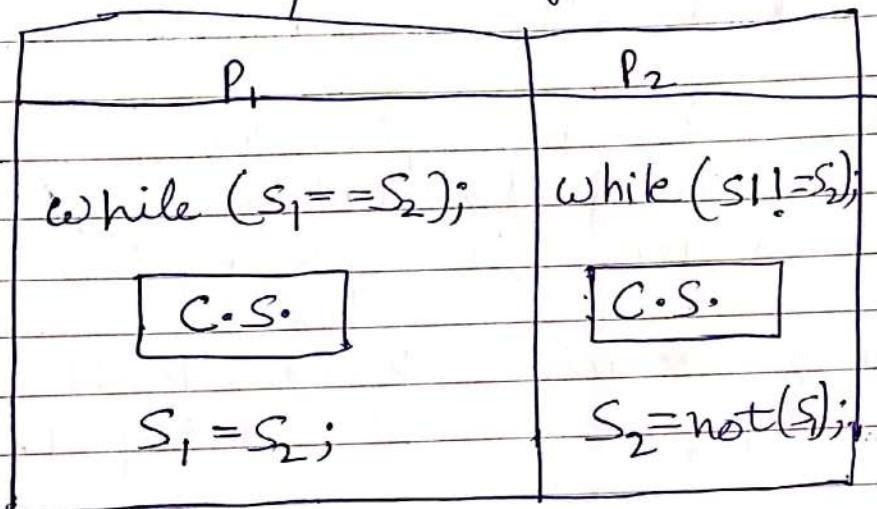
Q-15



S₁

(6)
S₂
180

doubt
Q.) Consider the two processes P_1 & P_2 , accessing their critical section. The initial values of shared boolean variables S_1, S_2 are randomly assigned.

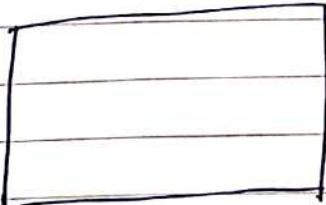


which of the following is true?

- a.) M.E is satisfied but NOT progress.
- b.) progress is satisfied but NOT M.E.
- c.) Neither M.E. NOR progress.
- d.) Both are satisfied.

Analysis:

$$\begin{aligned}S_1 &= 1 \\S_2 &= 0\end{aligned}$$



(a. Ans)

3) PETERSONS ALGORITHM:-

(2 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; → P0
    3. interested[process] = TRUE; → P1
    4. turn = process; → P0
    5. while (turn == process && interested[other] == TRUE);
}
```

C.S.

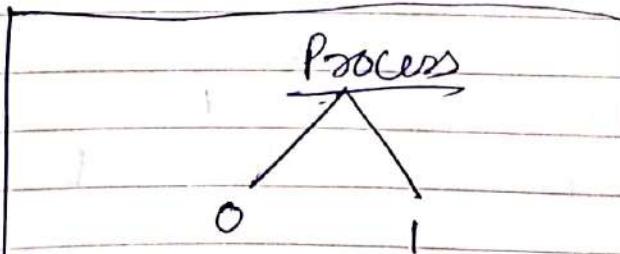
```
void leave-region(int Process)
{
```

```
    interested[process] = FALSE;
}
```

initially

interested[0] = FALSE;

interested[1] = FALSE;



"turn" is a shared variable used by both the process p_0 & p_1 .

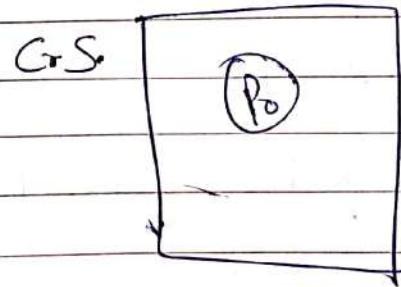
(found robin)

Analysis ① : Initially

interested[0] = FALSE;^{TRUE}
 interested[1] = FALSE;

turn = 0 |

Process 'P ₀ '	Process 'P ₁ '
other = 0	other = 0



Analysis ② : initially
~~interested[0]~~ false
 interested[0] = FALSE;
 interested[1] = FALSE; True

turn = 0 |

Process P ₀	Process P ₁
other = 1	other = 0

- 1.) M.E. is satisfied.
- 2.) Progress is satisfied.
- 3.) Bounded waiting is satisfied.

→ It is a correct solution.

We have proved that all the three conditions are satisfied in the above solution, hence it is consider to be correct solution & it is properly providing synchronization to the processes, to access the common Critical section.

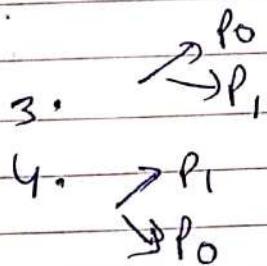
Doubt

$turn = 0;$

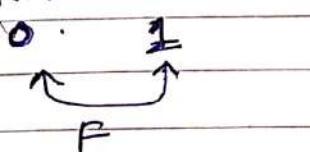
Q1.) Assume that both the processes P_0 & P_1 are trying to enter into critical section at the same time, then which process will enter into critical section first?

- a.) The process which executes statement '2' first.
- b.) The process which executes statement '3' first;
- c.) The process which executes statement '2' first;
- d.) We can not say.

(Solve)



Turn $\neq 0$. while (turn == process & & i == - -);



⇒ HARDWARE TYPE SOLUTION:-

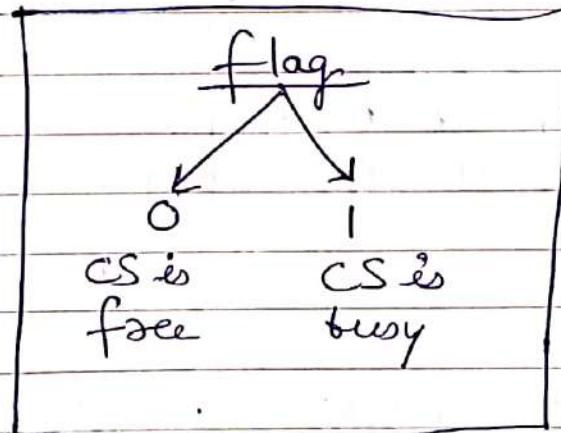
Q7

1) TSL instruction set:-

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

Entry section:-

- I. TSL R_i, m[flag]
- II. ~~BOP~~ CMP R_i, #0
- III. JNZ to step I
- IV. C.S.
- V. Store M[flag], #0.



⇒ Analysis :-

$$\text{flag} = \emptyset, X, 1$$

P₁ → I

R₁ [0]

P₁ → II

(P₁)

P₁ → III

P₁ → IV

C.S.

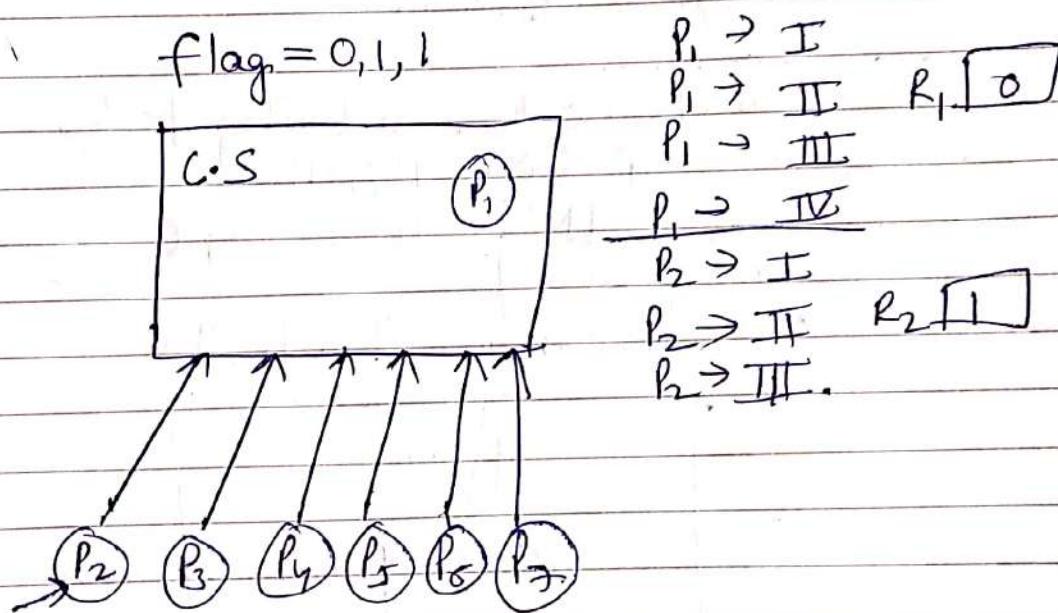
P₂ → I

R₂ [1]

P₂ → II

- 1.) Mutual Exclusion is satisfied.
- 2.) Progress is satisfied.
- 3.) Bounded waiting is not satisfied.

Analysis \rightarrow (Bounded waiting's)



Bounded waiting on Round Robin :- can be not satisfied.

$$\text{Flag} = 0, 1, 0$$

Ready Queue = $P_1, P_2, P_3, P_1, P_4, P_5, P_2, P_6, P_7, P_3$

- can going to indefinite waiting (starvation)

NOTE:- (1.) If some process is in the CS, then all the other processes, which are trying to enter into critical section will be repeatedly checking for I, II and III instructions.

(2.) These processes are busy in checking those instructions and waiting to enter into critical section, this is called as a "Busy waiting".

(3) If the "Busy waiting", we are wasting the CPU cycles (time).

Busy waiting is also called as "spin lock".

(3.) To avoid the problem of busy waiting, the concept of "Semaphore" will be used.

Table

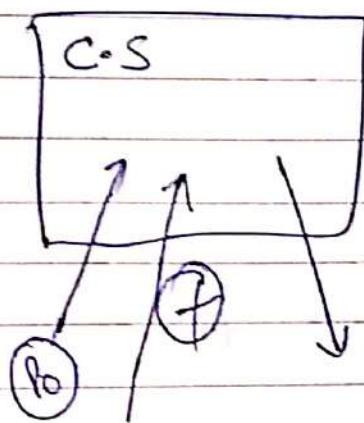
Solution	M · E	Progress	bounded waiting
lock variable	✗	✓	✗
Strict alternation (or) Dekkers algorithm	✓	✗	✓
Petersons algorithm	✓	✓	✓
TSL instruction set	✓	✓	✗

Progress

91

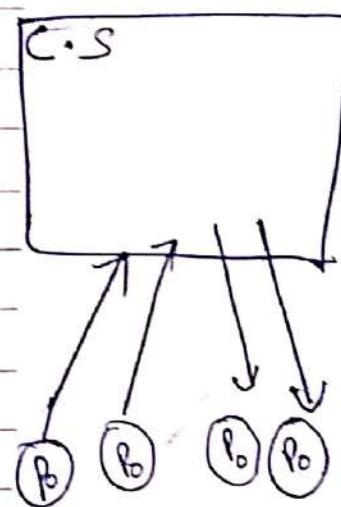
Strict alternation

processes $\Rightarrow (P_0, P_1)$
progress \otimes



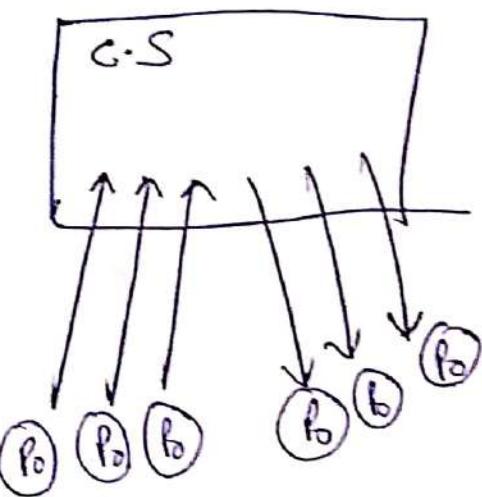
Petersons algo

processes $\Rightarrow (P_0, P_1)$
progress \ominus



TSL instruction set

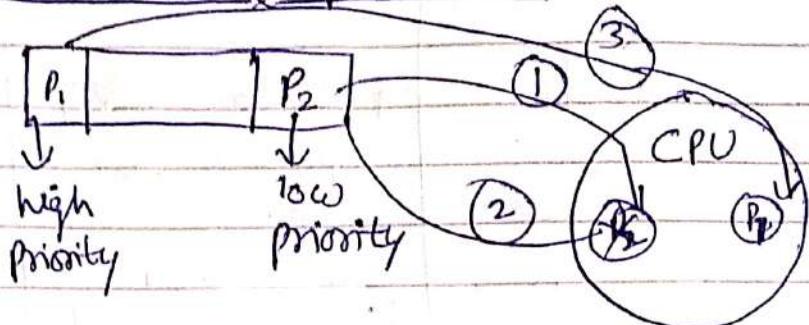
processes $\Rightarrow (P_0, P_1, P_2, P_3, \dots)$
progress \checkmark



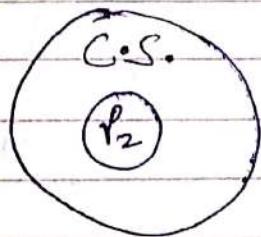
first check deadlock, then progress.

Priority Inversion problem:-

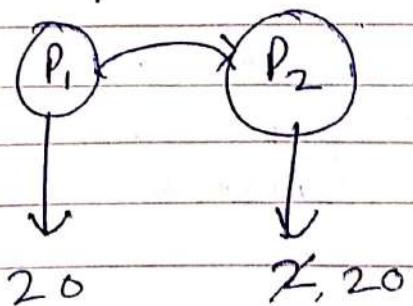
Ready Queue



live lock
P₂ → Ready
P₁ → run



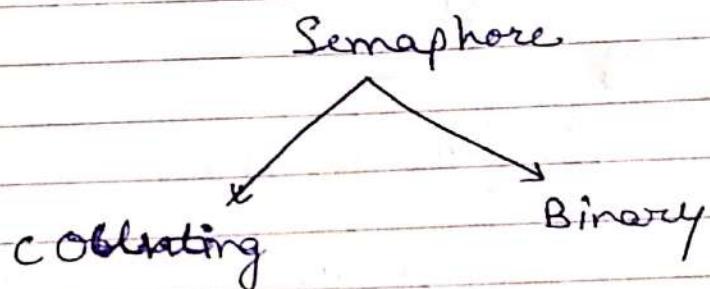
solution:- priority inheritance



changing the priority, from lowest to highest

SEMAPHORE:- Semaphore is an integer variable used by the processes in a mutual exclusive manner to achieve synchronization.

- The improper usage of semaphore will also give the wrong results.
- Semaphore is categorized into two types
 - 1.) ~~obliging~~ ^{Counting} semaphore
 - 2.) Binary semaphore



- The two different operations performed on a semaphore variable.

- 1.) Down(); or wait(); or p();
- 2.) Up(); or signal(); or V() or release();

COUNTING SEMAPHORE:-

Down (Semaphore S)

{
 s.value = s.value - 1;
 if (s.value < 0)
 {

 Block the process and
 place its PCB in the
 suspended list();

}

}

Up (semaphore S)

{
 s.value = s.value + 1;
 if (s.value >= 0)
 {

 Select a process from
 the suspended list and
 wakeup();

}

}

NOTE:- After performing down operation if the process is getting suspended, then it is called as a unsuccessful down operation.

→ If it is a successful down operation, then the process will not continue further execution in code.

$$\boxed{q \geq 1}$$

Q2
95

- After performing down operation, if the process is not getting suspended, then it is called as a successful down operation.
- If it is a successful down operation, then only the process will continue further execution in the code.
- The initial positive value of the semaphore indicates No of successful down operations we can perform.
- The down operation of Counting semaphore will be successful only if when initial value of semaphore

$$\boxed{S \geq 1.}$$

~~Up operation~~ → The process performing up operation will definitely continue the further execution.

- There is no unsuccessful UP operation and UP operation is always successful.

Q1.) Consider a system where initial value of the Counting semaphore $s = +17$, then various semaphore operations like $20P$, $14V$, $6P$, $8V$, $3P$ are performed. Then what is the final value of the semaphore.

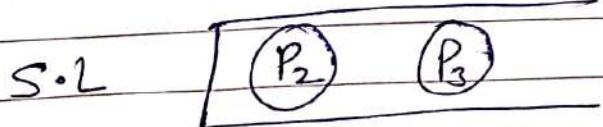
Solve.) $s = +17 - 20 + 14 - 6 + 8 - 3$.

$$s = 10.$$

NOTE:- → The negative value of the counting semaphores indicates no of processes (blocked / suspended lists.)

Example:- $s = +1, 0, -1, -2$

P_1	P_2	P_3
✓	✗	✗



Crack 2016)
Numerical Type

- (Q.) Consider the counting semaphore S . The various semaphore operations like $20P, 12V$ are performed. Then what is the largest initial value of semaphore S , so that 1 process will remain in a blocked lists.

Ans.)

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

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

$$\boxed{S = 7}$$

BINARY SEMAPHORE :-

Down (Semaphore S)

```
{  
    if (S.value == 1)  
        S.value = 0;  
    else
```

```
    {  
        Block the process and  
        place its pcb in the  
        suspended list();  
    }
```

UP (Semaphore S)

```
{  
    if (Suspended list is empty)  
        S.value = 1;
```

else

```
{  
    Select the process from  
    suspended list and wakeup();  
}
```

}

NOTE :- 1.) After performing down operation if the process is getting suspended then it is called as a unsuccessful down operation. If it is a unsuccessful down operation, the process will not continue further execution of

- After performing down operation, if the process is not suspended, then it is called as successful down operation.
- If it is a successful operation then only process will continue further execution in the code.

Down operation → The down operation in the binary operation will be successful, only when initial value of semaphore is '1'.

Up operation :- There is no unsuccessful up operation, up operation is always successful.

→ The process performing up operation will definitely continue the execution.

$$S = \phi, \phi, 1, 0, 1$$

P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇
P	P	P	V V	V V	V V	
✓	✗	✗	✓	✓	✓	✓

S.O.

(P₂) (P₃)

Q1) Consider the binary semaphore variable $S=1$. If the various semaphore operations like $gP, 14V, 6P, 8V, 3P, 2V$ are performed. Then what is the final value of the binary semaphore?

Ans.)

$$S = \begin{matrix} 1/0, 1, 0, 1, 0, \\ gP, 14V, 6P, 8V, 3P, 2V \end{matrix}$$

S.L 8φ, 5φ 20

final answer is 0.

NOTE:-

- 1.) Every semaphore variable will have its own suspended list.
- 2.) The down and Up operations are atomic.
- 3.) When more than one process is in the suspended list, then every time when one process will wake up and that will be based on FIFO.
(No preemption before or after down)
- 4.) If two or more processes are in the suspended list and there is no other process to wake up. These processes then those processes are said to be involved in the deadlock.

5) when the process is in the suspended list, the process will reside in the ready state, but it will not participate in the scheduling.

(Q1) Each process P_i , $i = 1$ to 9 executes the following code.

while (true)

{

p (mutex);

C.S.

v (mutex);

}

The process ' P_{10} ' executes the following code

while (true)

{

v (mutex);

C.S.

v (mutex);

}

6(i) [p (mutex)]
6(ii) v (mutex)

6(iii) [v (mutex)]
6(iv) p (mutex)

what is the max no. of processes, that may present inside the critical section at any point of time?

NOTE: The initial value of binary semaphore mutex = 1;

a.) 2.
(option b in case of p(mutex);)
b.) 3.
c.) 9.
(d.) 10.

e.) 1 \Rightarrow (iii) Ans

Q.) Consider the two concurrent processes 'P' and 'Q' executing their respective codes.

process 'P' code

while (true)

{

(w): —

print ('0');

print ('0');

(x): —

}

process 'Q' code

while (true)

{

(y): —

print ('1');

print ('1');

(z): —

}

(i) What should be the binary semaphore operations on w, x, y, z respectively, and what must be the initial values of binary semaphore 'S' and 'T' in order to get print output always as 0 0 1 1 0 0 1 1 0 0 --- --- .

a.) $w = p(S), x = v(S), y = p(T), z = v(T), S = T = 1;$

b.) $w = p(S), x = v(T), y = p(T), z = v(S), S = T = 1;$

c.) $w = p(T), x = v(S), y = p(S), z = v(T), S = 1, T = 0;$

d.) $w = p(S), x = v(T), y = p(T), z = v(S), S = 1, T = 0;$
correct

010 or 0110
101 or 10001

Q.) Consider the \rightarrow (same as above)



(ii) which of the following will ensure that, the output string will never contain the substring of the form 01^n0 or 10^n1 where ' n ' is odd?

- a.) $x = V(s), w = p(t), y = P(t), z = V(t), s = t = 1;$
- b.) $x = V(s), w = p(t), y = P(s), z = V(t), s = t = 1;$
- c.) $x = V(s), w = p(s), y = P(s), z = V(s), s = 1;$
- d.) $x = P(t), w = v(s), y = p(s), z = V(t), s = t = 1;$

2009

Q17.)

(option-d)

A

$$L_1 : P(S_x);$$

$$L_2 : P(S_y);$$

C-S

$$V(S_x);$$

$$V(S_y);$$

?

B

$$L_3 : P(S_x)$$

$$L_4 : P(S_y)$$

C-S.

$$V(S_y);$$

$$V(S_x);$$

?

Analysis:-

$$S_x = 1, 0, 1$$

$$S_y = 1, 0, 1$$

$$S_L = S_x \boxed{B}$$

workbook

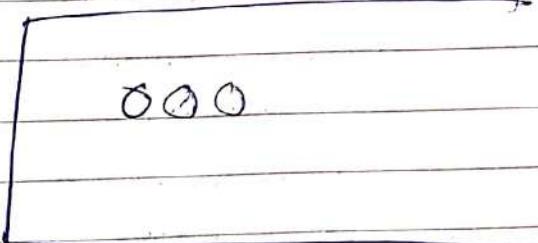
Q16.) Ans.)

Case 1:-

$$S_0 = 1 \cdot \emptyset \times 1 \text{ or } 0$$

$$S_1 = 0 \cdot 1 \times 1$$

$$S_2 = 0 \cdot 1 \times 1$$



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

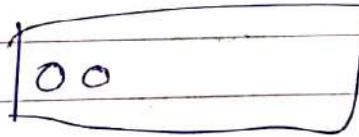
$$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_0$$

Case 2:-

$$S_0 = 1 \times 1 \cdot 1$$

$$S_1 = \emptyset \times 1$$

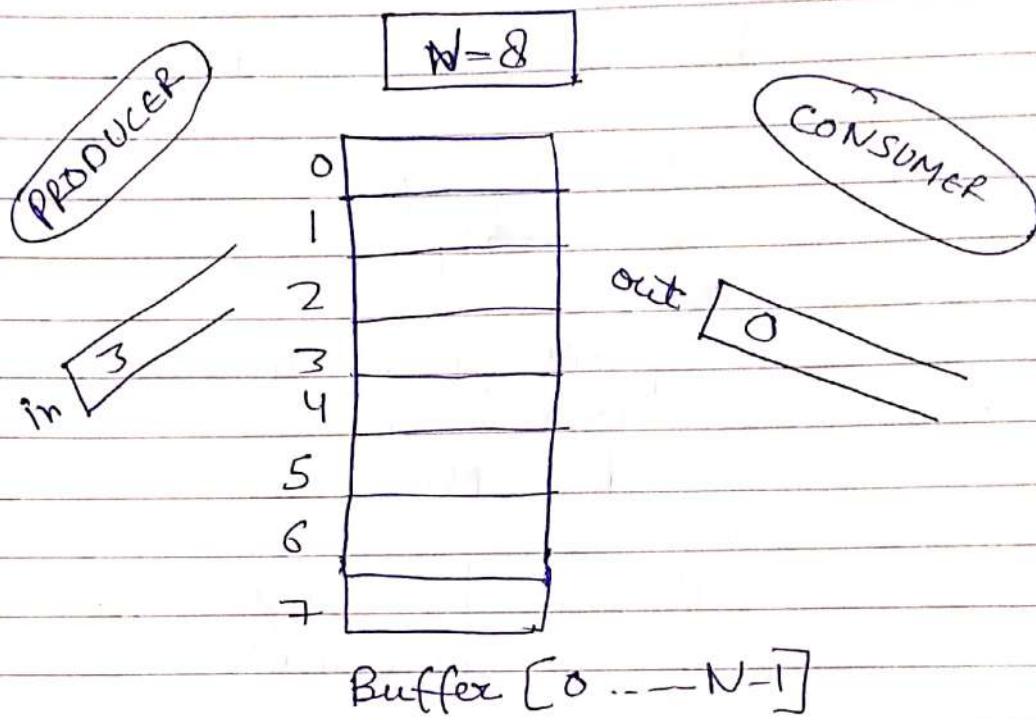
$$S_2 = \emptyset \otimes 1$$



(g. Ans.) At least twice

CLASSICAL PROBLEMS OF IPC:-

PRODUCERS- CONSUMERS with Semaphore :-



```
Semaphore mutex = 1;  
Semaphore Empty = N;  
Semaphore Full = 0;
```

```
void producer (void)  
{
```

```
    int itemp;  
    while (true)  
    {
```

```
        produce-item (itemp);  
        down (Empty);  
        down (Mutex);
```

```
        Buffer[in] = itemp;  
        in = (in+1) mod N;
```

```
        Up (Mutex);  
        Up (Full);
```

}

```
}  
void consumer (void)
```

```
{  
    int itemc;  
    while (true)  
    {  
        down (Full);  
        down (mutex);
```

```
    }  
    itemc = Buffer [out];
```

}

Up(mutex);
 Up(empty);
 process-item(itemc);
 }

- Mutex is binary semaphore used by producer and consumer in a mutual exclusive manner to access the buffer.
- Empty is a counting semaphore variable represents no. of slots empty in the buffer at any point of time.
- Full is a counting semaphore variable represents no of slots full in the buffer at any point of time.

Analysis:

$$\text{Mutex} = \begin{smallmatrix} 1 & 0 & 1 & 0 & 1 \end{smallmatrix}$$

$$\text{Empty} = \begin{smallmatrix} 3 & 1 & 5 & 4 \end{smallmatrix}$$

$$\text{Full} = \begin{smallmatrix} 2 & 4 & 3 \\ \hline 8 & 8 & 8 \end{smallmatrix}$$

Item P = A, B.

Item C = X, Y.

Q.) what happens if we interchange

→ down(Empty);
down (mutex);

in the producer 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 items produced by the producers will be lost.
- (d.) It is possible for deadlock.

Analysis:-

$$\left[\begin{array}{l} \text{Mutex} = 1, 0 \\ \text{Empty} = 0, -1 \\ \text{Full} = 1, 1 \end{array} \right]$$

take producer as full.

Q2.) What happens if we interchange

↳ up (mutex);
↳ up (full);

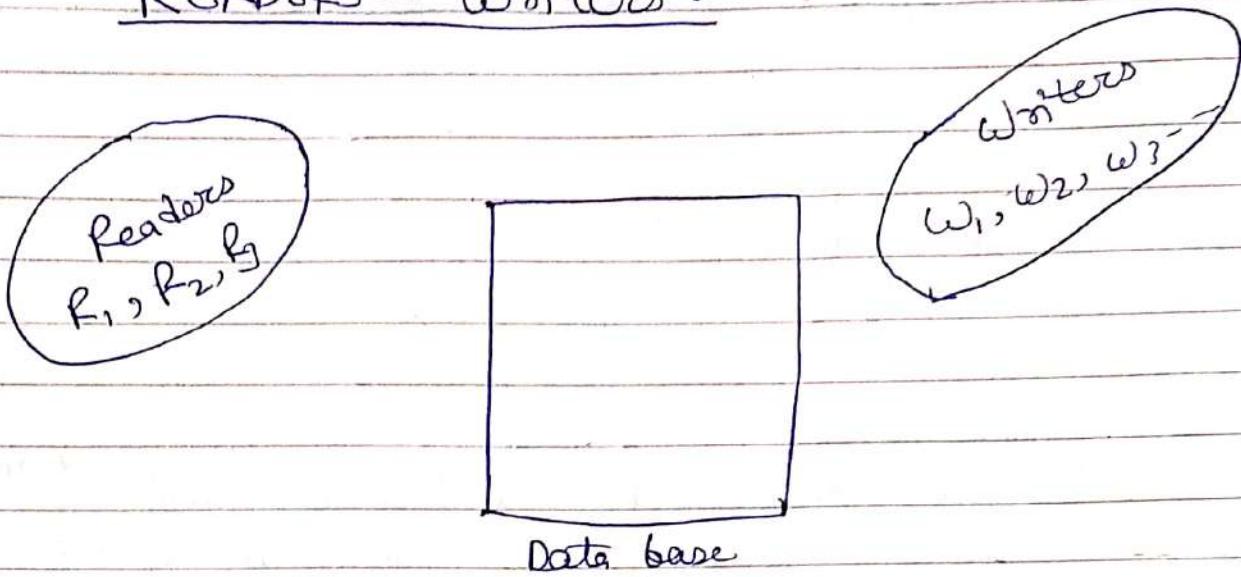
a.) No problem.

b.) both consumer & producer will access the buffer ^{at same} time.

c.) some of the items by the producer will be lost.

d.) it is possible for deadlock.

Readers - Writers :-



```
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);
```

}

3

```

void writer(void)
{
    while(true)
    {
        down(db);
        D.B.
        up(db);
    }
}

```

4 conditions to be followed :-

- [1.) R → W X]
- [2.) R → R ✓]
- [3.) W → R X]
- [4.) W → W X]

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

→ Mutex is a binary semaphore used by the reader's in a mutual exclusive manner.

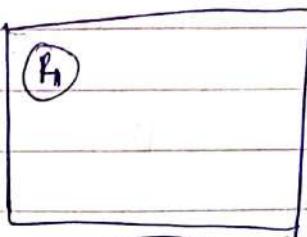
→ db is also a binary semaphore used by the reader's in a mutual exclusive manner.

Analysis 1 :-

$$RC = \emptyset, 1$$

$$\text{Mutex} = 1 \neq 1$$

$$db = 1 \ 0$$



(D.B.)

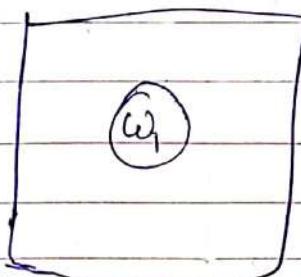
[for first 2 conditions]

Analysis 2 :-

$$RC = \emptyset$$

$$\text{Mutex} = 1$$

$$db = 1 \ 0$$



(D.B.)

1Q.) what happens if we interchange

\downarrow down (mutex); \swarrow
 \downarrow ~~down~~ $xc = xc + 1;$ \searrow

in a reader's code?

- a.) No problem, the solution still works correct.
- b.) Multiple readers are not allowed into database.
- c.) both reader and writer will enter into database at the same time.
- d.) It is possible for deadlock.

2Q.) what happens if we interchange

\nearrow Preemption.

\downarrow if ($xc == 1$) down (d6); \swarrow
 \downarrow up (mutex); \searrow

(Preemption before if)

in the reader's code.

\nearrow some

- a.) No problem, the solution will work correct.

- b.) Multiple readers are not allowed into database.

- c.) both reader and writer will enter into database at the same time.

- d.) It is possible for deadlock.

3. (Q) what happens if we interchange

→ down(mutex);
→ rc = rc - 1;

in the reader's code.

- a.) No, The problem will still work properly.
- b.) Multiple readers are not allowed into database.
- c.) both reader and writer will enter into database at the same time.
- d.) It is possible for deadlock.

Solve:)

(2/02/19)

Q-95

19.)

mutex =

 $r=0$; 1 $w=0$; 2

[] []

15]

Analysis :-

 $r=\emptyset$ $w=0$ mutex = $\neq \emptyset$

D.B.

⊗

up(mutex); - 2

⊗

 $r \geq 1$ or $w = 1$ - 3

⊗

→ $r=0, w=0, \text{mutex} = 1$

⊗ right answer

(v) (d) Ans:

Q.) Consider the following code used by classical readers & writers.

which of the following is true?

a.) The solution is correct & it is properly providing the synchronization to classical readers & writers.

b.) Multiple writers are allowed to enter into database at the same time.

c.) Both Reader & writers are allowed to enter into database at the same time.

d.) It is possible for deadlock.

int $rc = 0$;

int $wc = 0$;

Semaphore $mutex = 1$;

Semaphore $r = 1$;

Semaphore $db = 1$;

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);

D.B.

down(mutex);

$rc = rc - 1;$

if ($rc == 0$) up(d6);
 } } up(mutex);

void writer(void)
 { while (TRUE)
 {

down(r);

$wc = wc + 1;$

down(d6);

D.B.

up(d6);

$wc = wc - 1;$

up(r);

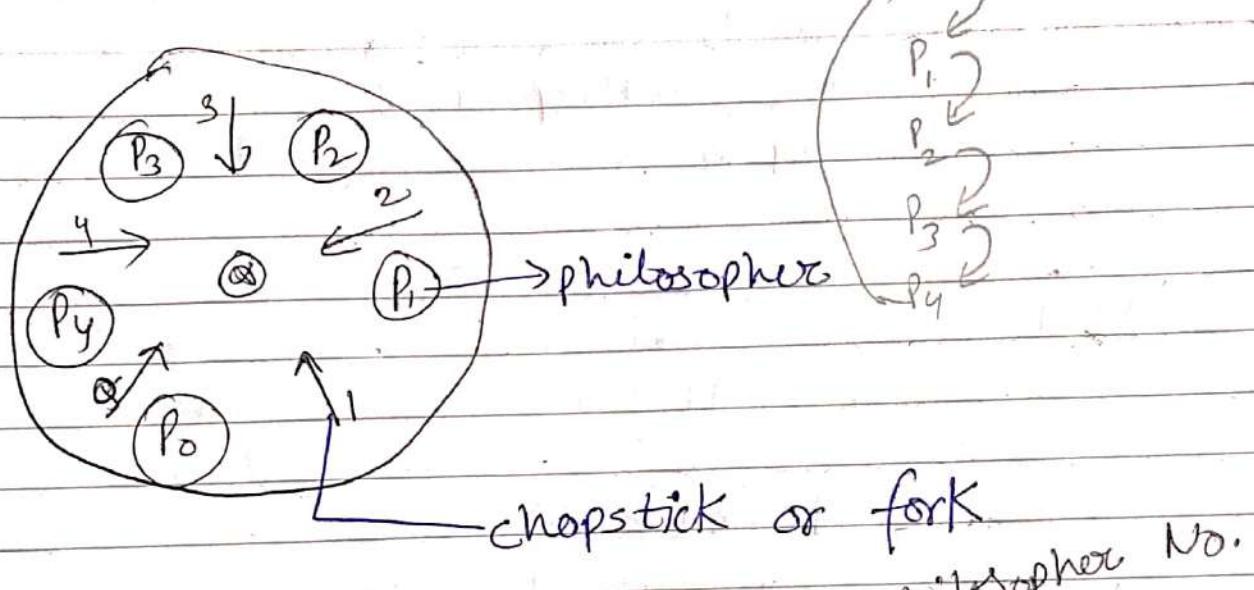
}

}

(d Ans) deadlock.

Dining philosophers

N=5



void philosopher (int i) {
 philosopher No.

while (true)

thinking();

take_fork(i); // take left fork

take_fork((i+1) % N); // take right fork

eat();

put_fork(i); // put left fork back

put_fork((i+1) % N); // put right fork back

}

3

If all the philosophers are hungry at the same time, then everybody will take their left fork first and when they tried to attempt to right fork, then nobody will get the right fork and all the philosophers will wait on each other for the Right fork and they will going to deadlock.

Solution with the help of Semaphore.

```
# define N 5
```

```
# define THINKING 0
```

```
# define HUNGRY 1
```

```
# define EATING 2 → left philosopher
```

```
# define LEFT  $\lceil (i+N-1) \% N \rceil$ 
```

```
# define RIGHT  $(i+1) \% N \rceil$  → right philosopher
```

```
Semaphore mutex = 1;
```

```
Semaphore S[N]; // all S[i]'s are initialized to '0' ;
```

```
int state[N];
```

// an array to keep track of every philosopher's state.

```
void philosopher ( int i ) → philosopher No
```

```
{
```

```
while ( true )
```

```
{
```

```
thinking();
```

```
take-forks(i);
```

```
eat();
```

}

}
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 ;
 test (LEFT);
 test (RIGHT);
 up (mutex);

}

void test (int i)

{
 if (state [i] == HUNGRY && state [LEFT]
 != EATING &&
 state [RIGHT] != EATING)

{
 state [i] = EATING ;
 Up (s [i]);

}

}

- 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. Initially all the philosophers will be in the thinking state.

- Analysis :-

$\text{Mutex} = \emptyset$

$P_0 = T$	$S[0] = 0$
$P_1 = T$	$S[1] = 0$
$\Rightarrow P_2 = T \times E$	$S[2] = \emptyset \times 0$
$P_3 = T$	$S[3] = 0$
$P_4 = T$	$S[4] = 0$

Q1) Assume that philosopher P_1 & P_3 are in eating state. Then philosopher P_2 is also trying to go to eating state. Then which statement in the above code is controlling the philosopher?

- a.) down(mutex);
- b.) If condition
- c.) test function
- d.) down(s[i]);

Q2) what happens if we interchange

→ up(mutex);
↳ down(s[i]); in the takefork()?

- a.) No problem the solution still works correct.
- b.) More than two philosophers can go into eating state at the same time.
- c.) It is possible for deadlock.
- d.) None of the above.

Now

Q.) Assume that P_1 and P_3 are in the eating state, then Philosopher P_2 is also trying to go to eating state, then it will be suspended. Then what is the procedure and how philosopher P_2 will go into eating state?

GATE

Q) Let $p[0] \dots p[4]$ be the processes and $m[0] \dots m[4]$ be the binary semaphore mutexes all are initialized to 1. Each process ' p_i ' executes the following code.

$\text{wait}(m[i]);$ ~~Preempted~~
 $\text{wait}(m[(i+1) \% 4]);$

C.S.

$\text{Signal}(m[i]);$
 $\text{Signal}(m[(i+1) \% 4]);$

→ Consider the following statements

I. M.E. is Satisfied.

II. M.E is NOT Satisfied.

III. It is possible for deadlock.

Which of the above are TRUE?

a) only I

b) only II

c) only I, III

d) only II, III

P₀ P₁

C.S.

Analysis:

$$P_0 = \cancel{m_0, m_1}$$

$$m_0 = 1$$

$$P_1 = m_1, m_2$$

$$m_1 = 1$$

$$P_2 = \cancel{m_2, m_3}$$

$$m_2 = 1$$

$$P_3 = \cancel{m_3, m_0}$$

$$m_3 = 1$$

Q.) Consider the following code used by the processes to access the common critical section.

Shared boolean $LOCK = FALSE;$

boolean $key;$

do

{

$key = TRUE;$

while ($key == TRUE$)

$Swap(key, lock);$

C.S.

$lock = FALSE;$

} while ($TRUE$);

NOTE: $Swap()$ function atomically swaps two variables by using call by reference.

=> Consider the following statements

I. M.E is satisfied

II. M.E is not satisfied

III. It is possible for deadlock.

Which of the above are TRUE ?

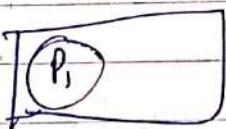
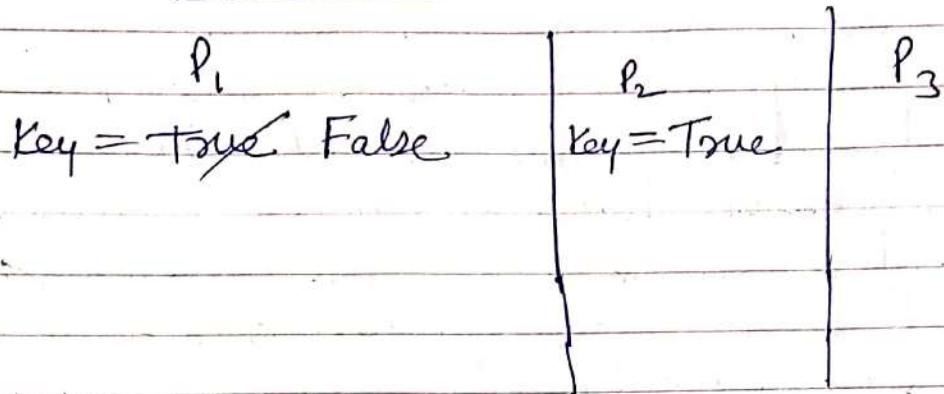
a.) Only I ✓

b.) Only II

c.) Only I, III

d.) Only II, III

LOCK = FALSE TRUE



C-S

MONITORS:-

- Monitors is programming language, compiler support type of solution to achieve synchronization.
- Monitors is collection of variables, condition variables and procedures combined together in a special kind of module or package.
- The processes running outside the monitor cannot directly access, the internal variables of the monitor but however they can call procedures of the monitor.
- Monitors has an important property that only one process can be active inside the monitor at any point of time.

Syntax :- ^{Keyword} ^{monitor example} { Name of the monitor }

```

    {
        variables;
        condition variables;
        procedure P1;
    }
    =

```

```
procedure P2
```

```
{
    =
}
```

CONDITION VARIABLES :-

Condition x, y;

The two different operations performed on the condition variables of the monitor:

- 1.) wait();
- 2.) signal();

1.) Wait() :- The process performing wait operation on many condition variable will be suspended and suspended process will be placed in the "block Queue" of respective condition variable.

2.) Signal() :- The process performing signal operation will have two cases

Signal

Block Queue
is empty

→ Signal has No effect
& Signal will be lost.

Block Queue is
NOT Empty

→ one process will be
removed from the block
Queue and any one
process will continue the

PRODUCERS - CONSUMERS WITH MONITORS :-

$N = 8$

Count = 3

0	X
1	Y
2	Z
3	
4	
5	
6	
7	

Buffer [0.....N-1]

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);

}

}

procedure producer

{

int itemp;

while (true)

{

produce-item (itemp);

producer-consumer-enter-item;

}

}

procedure consumer

{

int itemc;

while (true)

{ producer-consumer-remove-item;

process-item (itemc);

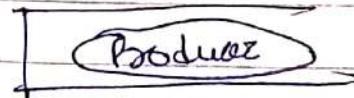
}

}

Analysis:-

Buffer is full :-

Block Queue of full

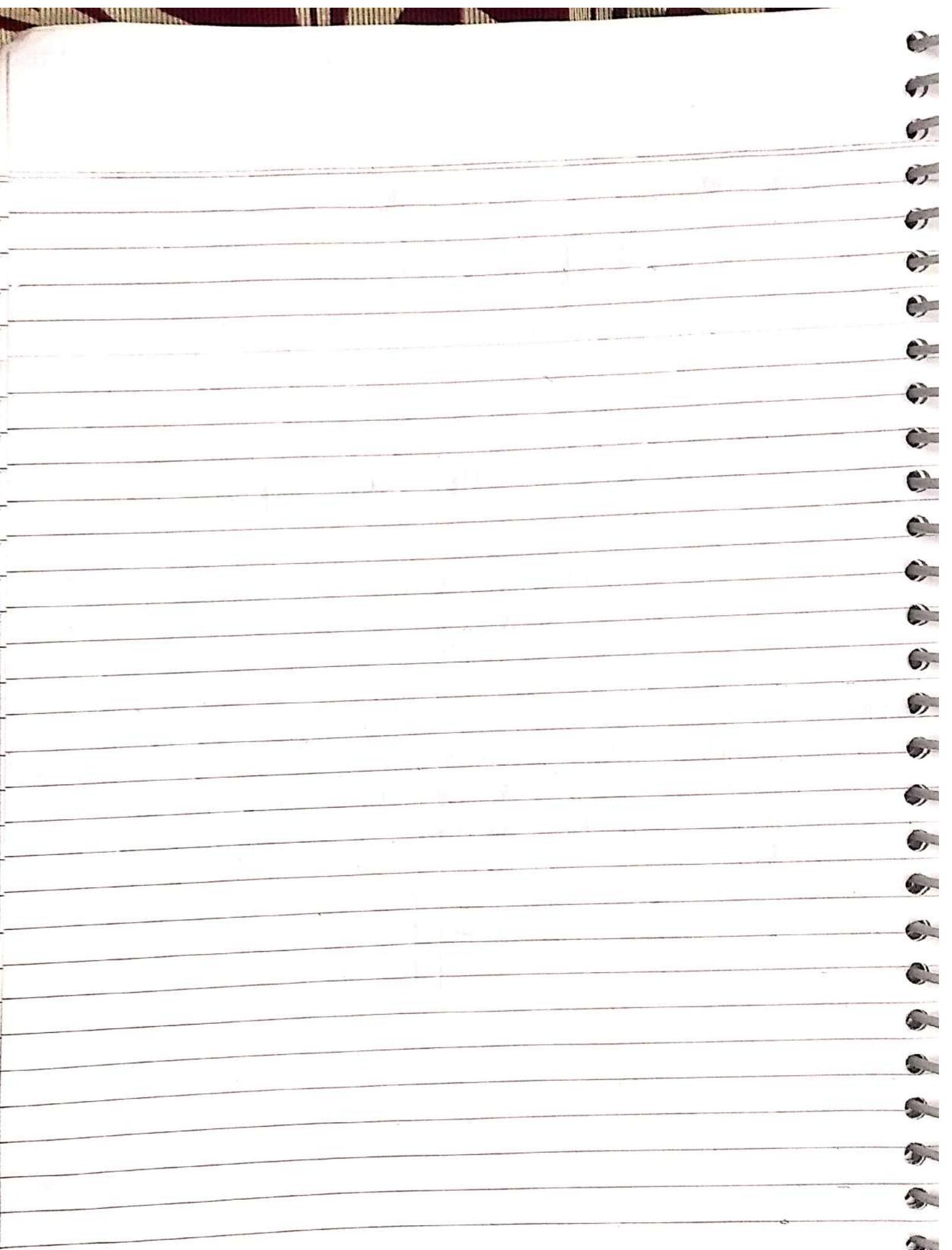


When Buffer is Empty :-

Block Queue of Empty



NOTE:- Even though we are using the count variable, there will be not be any inconsistency because inside the monitor only one process can be active at any point of time.



CONCURRENT PROGRAMMING:-

$S_1 : a = b + c;$

$S_2 : d = e * f;$

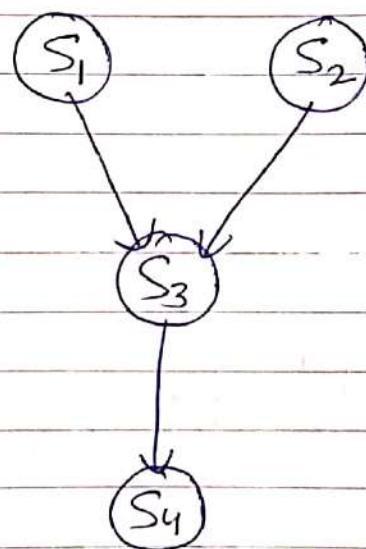
$S_3 : g = a / d;$

$S_4 : h = g * i;$

Read Set = {b, c, e, f, a, d, g, i};

Write Set = {a, d, g, h};

Precedence graph :-



NOTE :- Any two statements ' S_i ' and ' S_j ' can be executed concurrently or parallelly if they are following the below conditions.

$$1.) R(S_i) \cap W(S_j) = \emptyset.$$

$$2.) W(S_i) \cap R(S_j) = \emptyset.$$

$$3.) W(S_i) \cap W(S_j) = \emptyset.$$

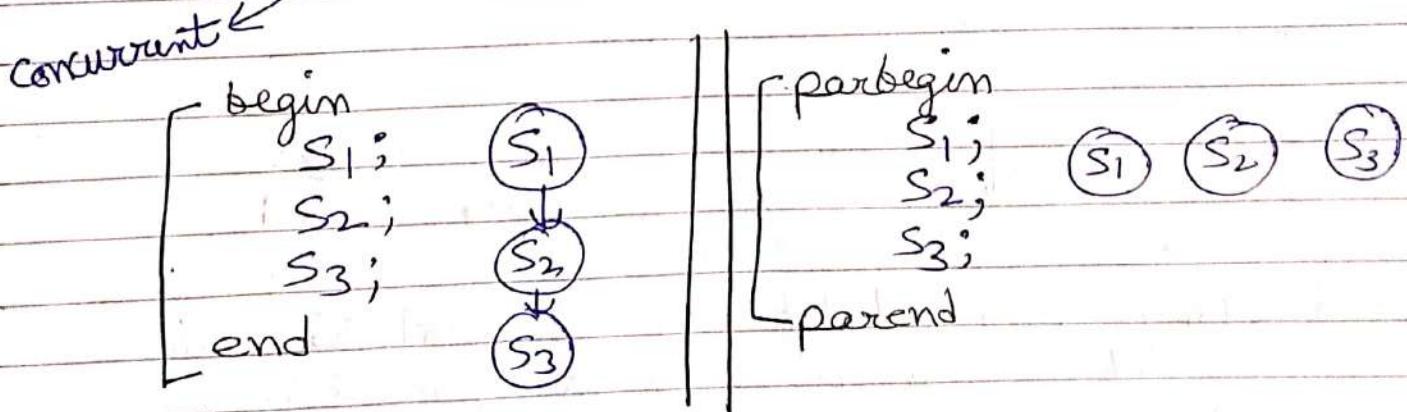
2.) → The real concurrent programming is possible only on the multiprocessor System.

3.) Concurrent has the three different meanings

- They can execute concurrently or parallelly.
- They don't have any dependency.
- * Any one can start first.

4.) The concurrent program will be written by using

parallel ←
[par begin - par end
(or)
cobegin - coend]



Eg(1): begin

$S_1;$

parbegin

begin

$S_2;$

$S_3;$

end

begin

$S_4;$

$S_5;$

end

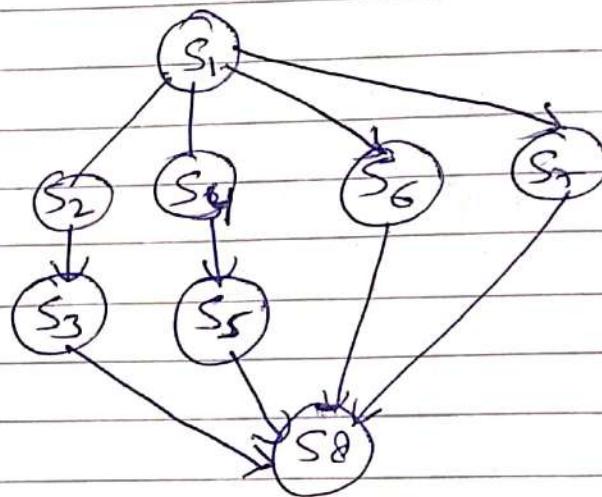
$S_6;$

$S_7;$

parend

$S_8;$

end



begin

$S_1;$

parbegin

begin $S_2;$

$S_4;$

parbegin

$S_5;$

$S_6;$

parend

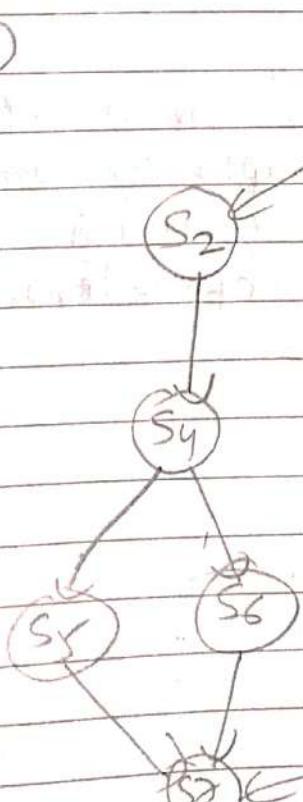
end

S_3

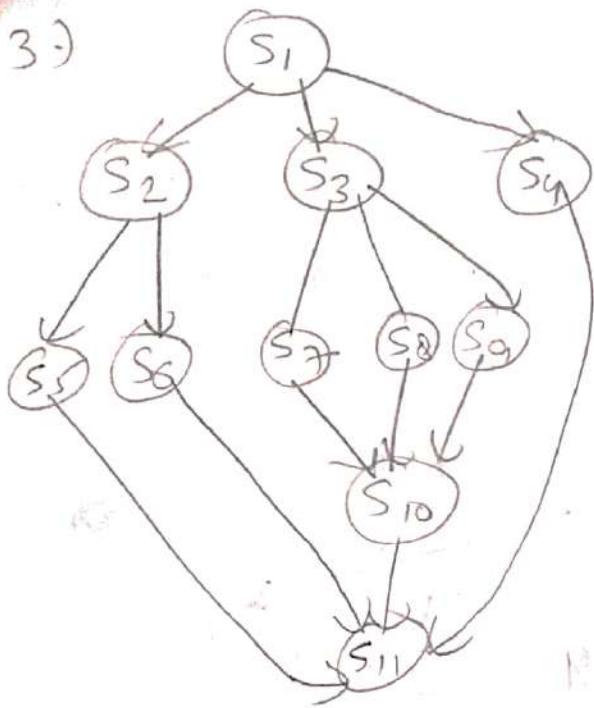
parend

$S_7;$

end



3)



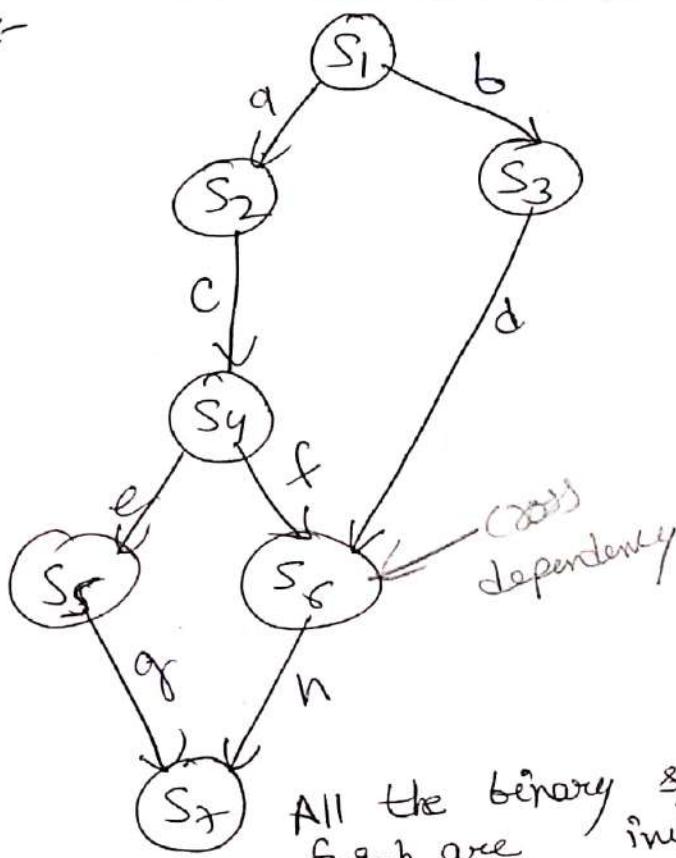
$S_1;$
~~parbegin~~
~~begin~~
 ~~$S_2;$~~
~~parbegin~~
 ~~$S_5;$~~
 ~~$S_6;$~~
~~parend;~~
~~end~~

~~begin~~
 ~~$S_3;$~~
~~parbegin~~
 ~~$S_7;$~~
 ~~$S_8;$~~
 ~~$S_9;$~~
~~parend~~
 ~~$S_{10};$~~
~~end~~
 ~~$S_4;$~~
~~parend~~

$S_{11};$

NOTE:- It is not possible to write the concurrent program for all the precedence graphs by using ~~parbegin~~ and ~~parend~~ but it is very much possible with the help of semaphore operations.

Ex:-



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

parbegin

begin S_1 , parbegin $v(a), v(b)$, parent, end;

begin $p(a), S_2, v(c)$, end;

begin $p(b), S_3, v(d)$, end;

begin $p(c), S_4$, parbegin $v(e), v(f)$, parent, end;

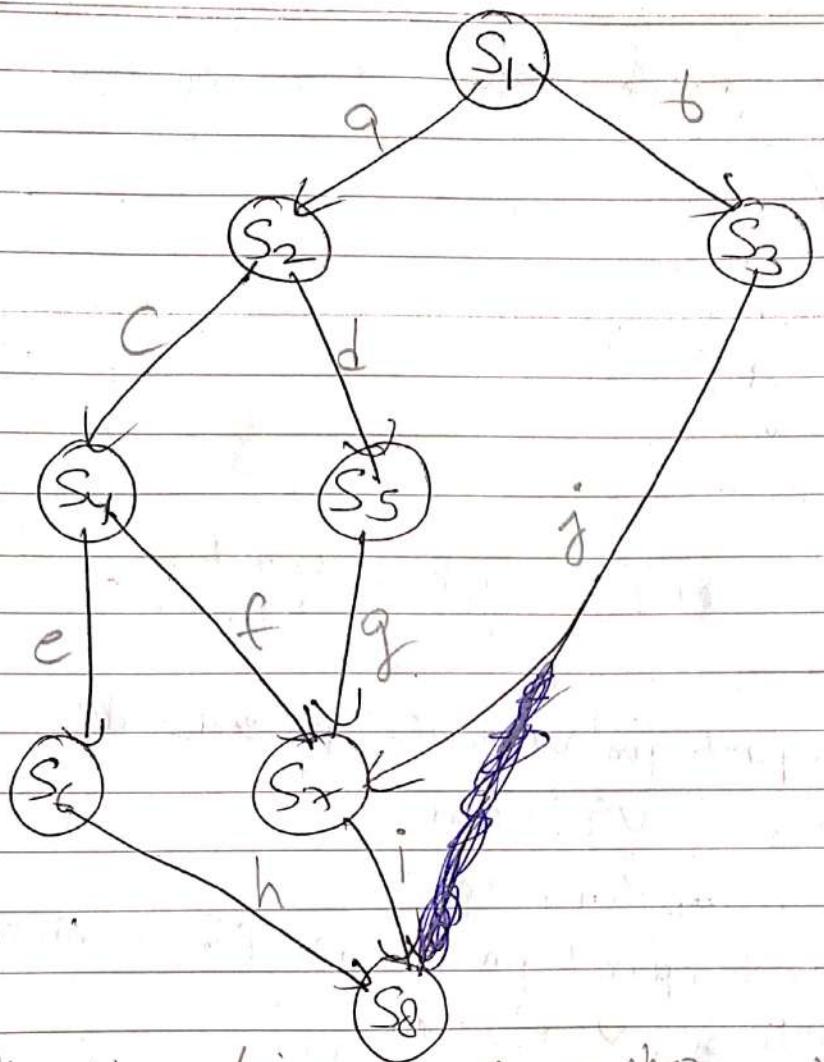
begin $p(e), S_5, v(g)$, end;

begin $p(d), p(f), S_6, v(h)$, end;

begin $p(g), p(h), S_7$, end;

parent

Q.)



All the binary semaphore variables
a, b, c, d, e, f, g, h, i, j is initialized to '0'.

parbegin

begin S₁, parbegin v(a), v(b), parend, end;

begin p(a), S₂, parbegin v(c), v(d),
parend, end;

begin p(c), S₄, parbegin v(e), v(f),
parend, end;

begin p(d), S₅, parbegin v(g), + 1,
end;

begin p(e), S₆; f; + 1, v(h), parend;
end;

begin p(f), P(g), P(j), S₇, + 1, v(i),
end;

begin p(b), S₃, v(j), end.

begin p(i), p(n), S₈, end.

Q.) Consider the following concurrent program:-

```

int x=0, y=0;
parbegin
  begin
    x=1;
    y = y + x;
  end
  begin
    y = 2;
    x = x + 3;
  end
end
  
```

what can be the final values of x and y after completion of above concurrent programs?

I. x=1, y=2

II. x=1, y=3

III. x=4, y=6

which of the following are possible?

- a.) only I, II
- b.) only II, III
- c.) only I, III
- d.) All I, II, III

P-91

(S.Q.) (6.Ans) only I, II, III.

2013

Q25)

P-96.)

option.)

- a.) - 2
- b.) - 1
- c.) + 1
- d.) + 2

$p(S): w, x$	$y, z \in p(S):$
I. load $r_i, M[x]$	I. load $r_i, M[x]$
II. INCR R_i	II. DECR R_i
III. Store $M[x], R_i$	III. Store $M[x], R_i$
$V(S);$	$V(S);$

w	x	y	z
$p(S);$	$p(S);$		
I. load $r_w, M[x]$	I. load $r_x, M[x]$	I. load $r_y, M[y]$	I. load $r_z, M[z]$
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 $M[x], R_z$
$V(S);$	$V(S);$	$V(S);$	$V(S);$
+ 1	+ 2	- 2	- 4

$$S=2$$

$$x = \underline{0, -2, -4, +1, +2}$$

$$\omega \rightarrow I$$

$$f(\omega) [0, 1]$$

$$\omega \rightarrow II$$

$$4 \rightarrow I, II, III$$

$$2 \rightarrow I, II, III$$

$$\omega \rightarrow III$$

$$x \rightarrow I, II, III$$

Q.) Consider the following concurrent program

```

int count = 0;
void tally()
{
    int i;
    for(i=1; i<=5; i++)
        count = count + 1;
}
main()
{
    parbegin
        tally();
        tally();
    parent
}
    
```

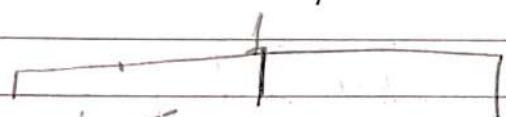
What can be the minimum final value of count after completion of both the functions of tally()

NOTE:- count = count + 1, will execute in '3' instructions like load, increment and store.

- I. load R_i , $m[\text{count}]$
 II. INCR R_i
 III. store $M[\text{count}], R_i$

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

Tally 1



$$\text{count} = 5$$

Tally 2



$$m(\text{count}) = 0$$

$$R_i = 0$$

$$R_i = 1 \quad \Leftarrow$$

$$m(\text{count}) = 1$$

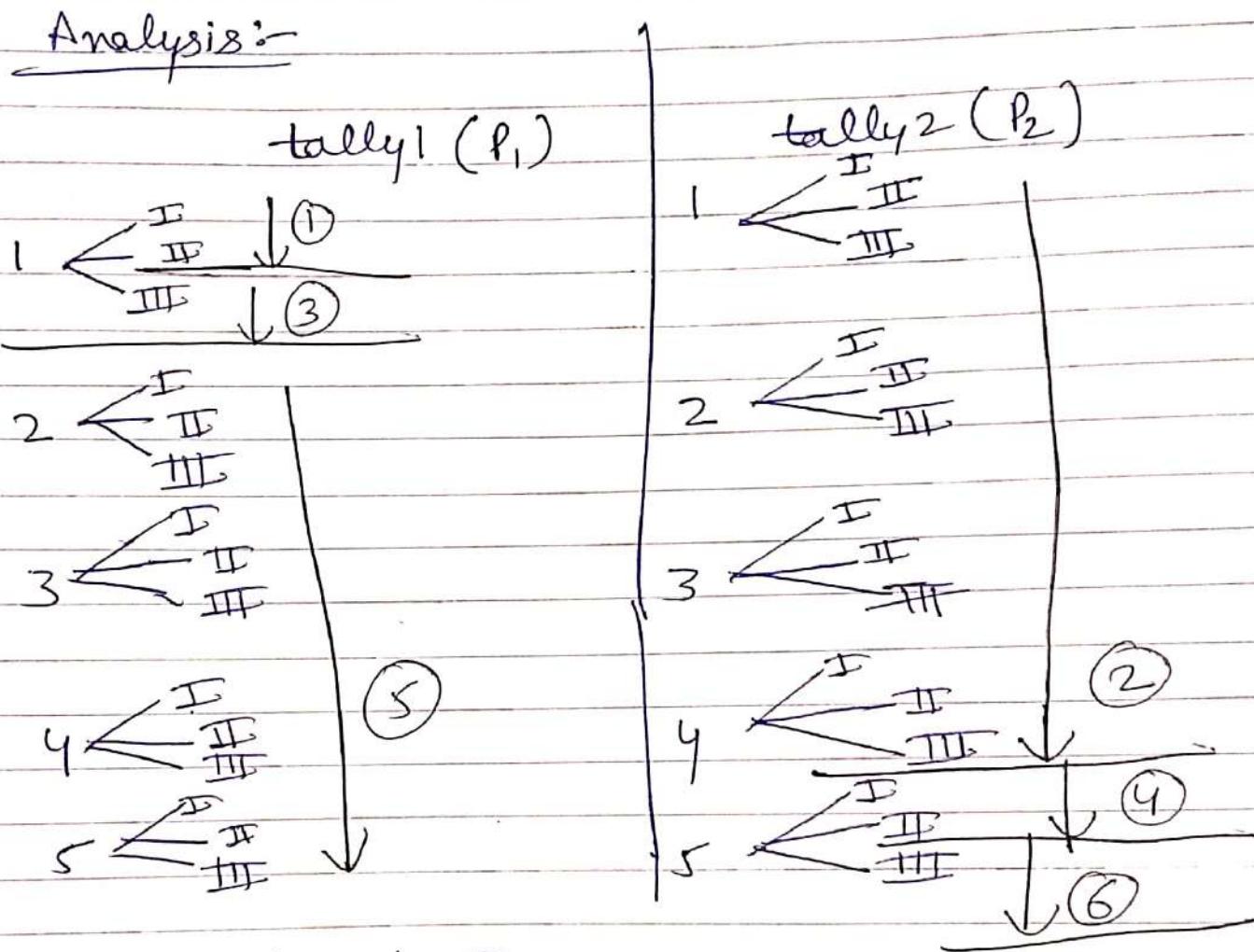
count

$$m(\text{count}) = 0$$

$$R_i = 0$$

$$m(\text{count}) = 1$$

Analysis :-



Count = 0

P₁ : 1st iteration → I

R₁ [∅, 1]

P₁ : 1st iteration → II

∅

P₂ : 1, 2, 3, 4 iterations

P₁ : 1st iteration → III

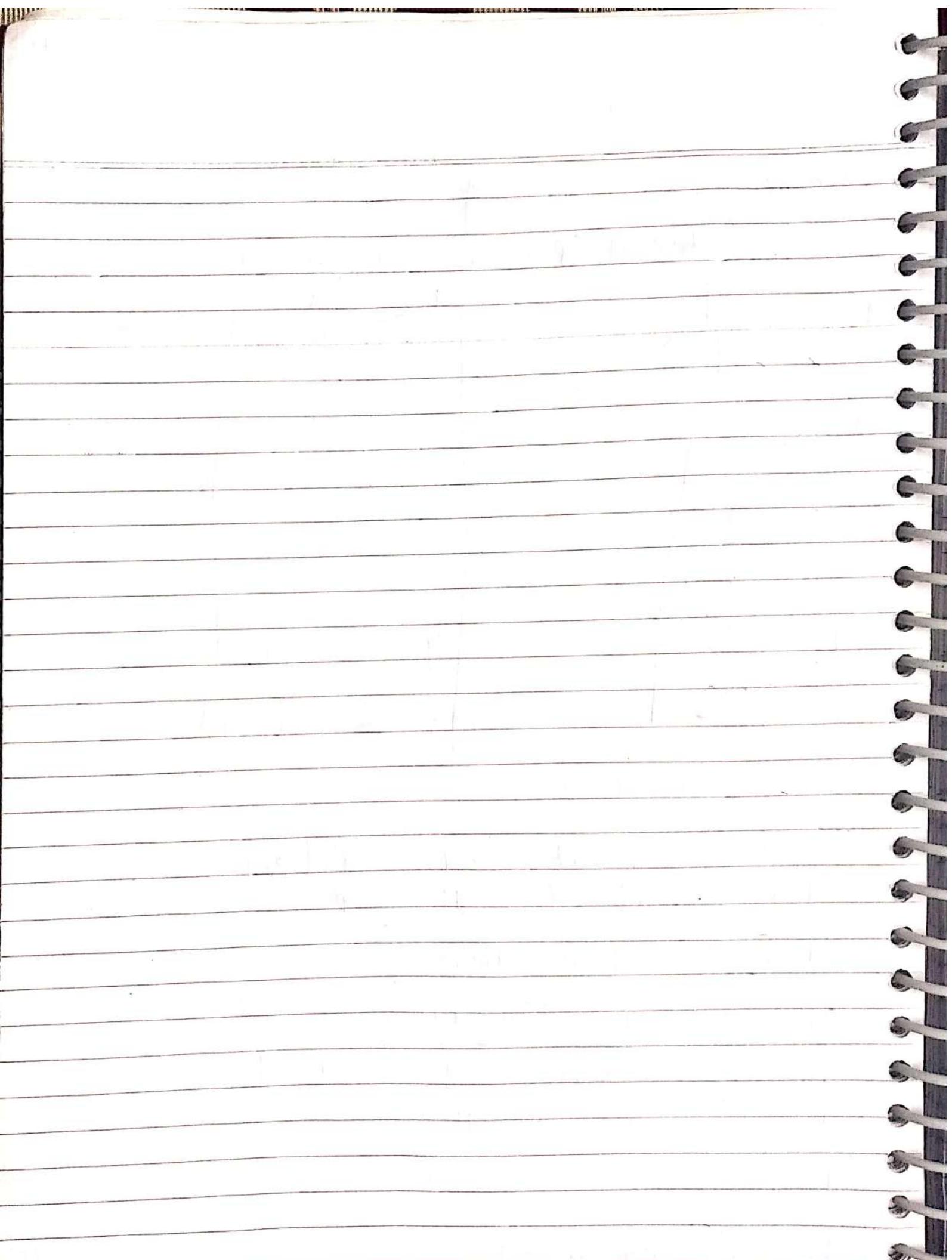
R₂ [x, 2]

P₂ : 5th iteration → I

P₂ : 5th iteration → II

P₁ : 2, 3, 4, 5 iterations

P₂ : 5th iteration → III



Short URL

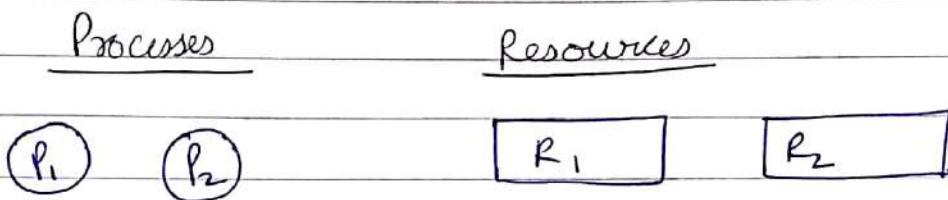
147

https://bit.ly/2TtKQno - C book

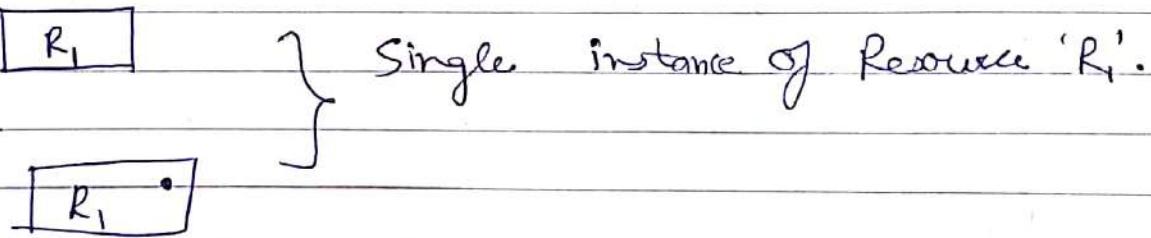
Deadlock :-

Def :- The two or more processes are waiting on some event to happen, which never happens then those processes are said to be involved in the deadlock.

Basis of Deadlocks :-



Resources with instances :-



R_2 : '2' instances of Resource ' R_2 '.

Requesting Edge :-

for one instance of Resource ' R_1 '.



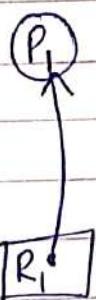
Process P_1 requesting

Deadlock



process 'P₂' is requesting, for '2' instances of resource 'R₂'.

ALLOCATION EDGE:-



'1' instance of resource 'R₁' allocated to process 'P₁'.



'2' instances of resource 'R₂' are allocated to process 'P₂'.

→ The resource request and resource allocation will be represented by using resource allocation graph.

$$RAG = G(V, E)$$

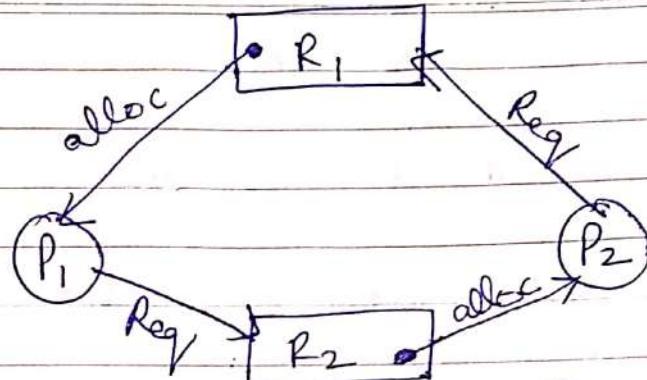
Processes,
Resources

Requesting,
Allocation

The Resource Request / Release Life Cycle:-

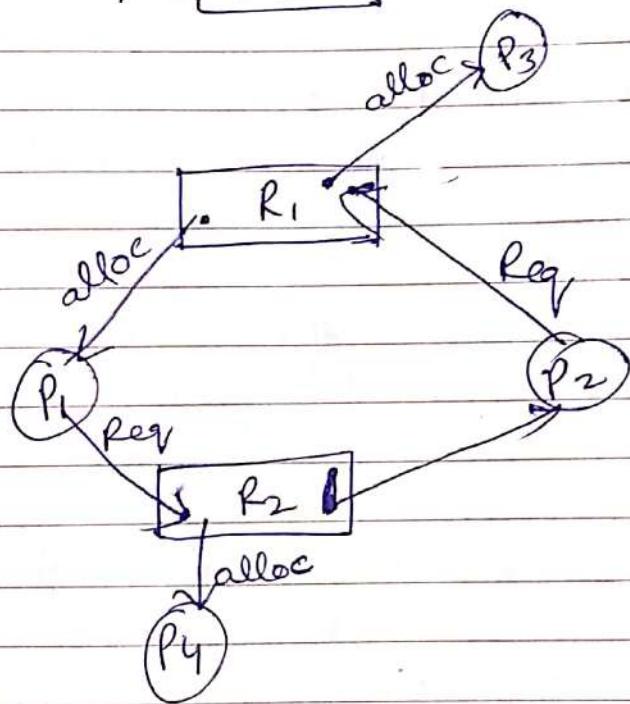
- 1.) The process will make the request for the resource.
- 2.) The operating system clearly validates the request of the process.
- 3.) If the request made by the process is valid, then the operating system will check for the availability of the resource.
- 4.) If the resource is freely available, then it will be allocated to the process, otherwise process has to wait.
- 5.) If all the resources required further start of execution are allocated then the process will go into execution.
- (6.) Once the execution of the process is completed then it will release all the resources.

Eg:



Deadlock exists in the RAG.

Eg. 2:



No deadlock exists in a RAG.

Q.) Consider a system which has n processes, 6 → Tape drives. Each process requires 2 tape drives to complete their execution. Then what is the maximum value of n, which ensures deadlock free execution?

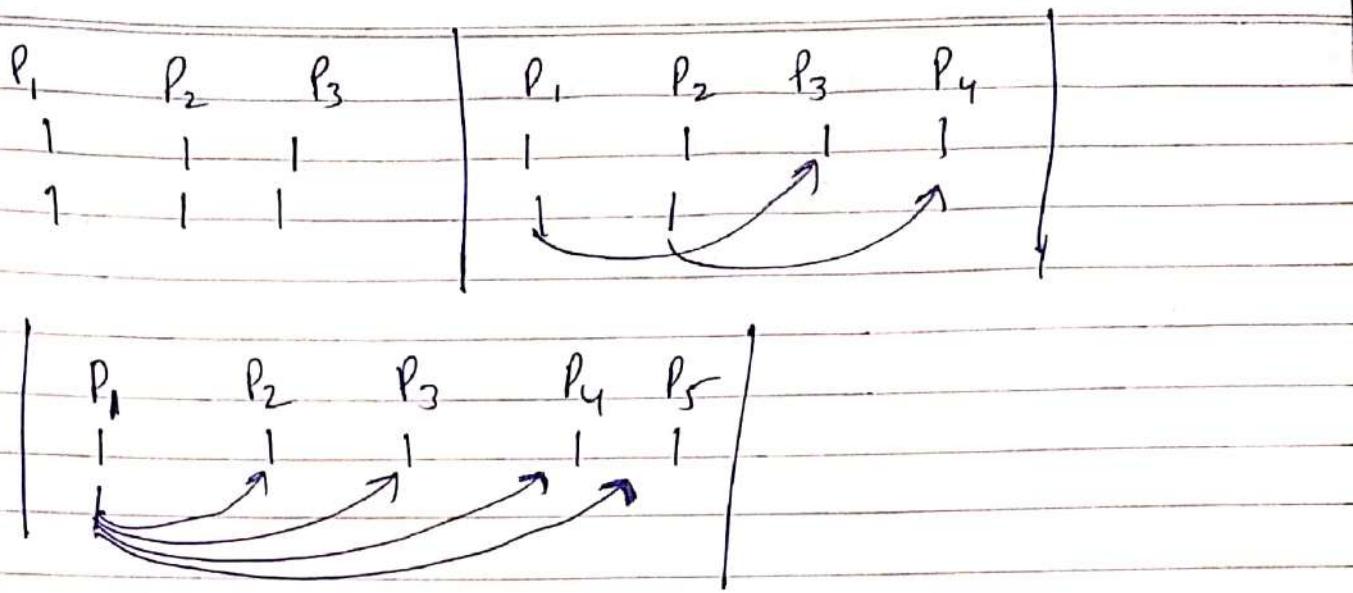
a.) 2

b.) 3

c.) 4

d.) 5

- 1



(Q.) Consider a system which has 3 processes & each process requires 2 resources to complete their execution. Then what is the minimum no. of resources are required to ensure deadlock free execution?

- a.) 2
- b.) 3
- c.) 4,
- d.) 5

P ₁	P ₂	P ₃
1	1	1
1	1	1

$$\begin{aligned}
 & \boxed{n+m-1} \\
 & = 3+2-1 \\
 & = 4
 \end{aligned}$$

1 + 3
3 4

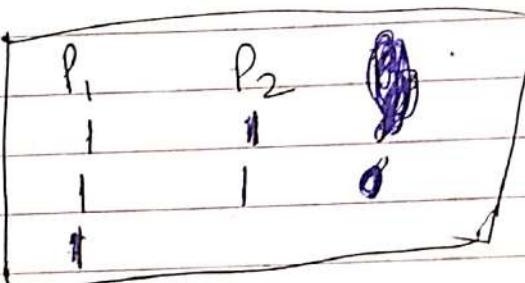
Q.) Consider a system which has n processes and 6 resources. Each process required 3 resources to complete there execution. What is the maximum value^{of process} to ensure deadlock free execution?

a.) 2

b.) 3

c.) 4

d.) 5



(a.) 2 .

Q.) Consider a system which has 3 processes P_1 , P_2 and P_3 . The peak demand of each process is 5, 9, 13 respectively for the resource 'R'. what is minimum no. of resources required to ensure deadlock free execution?

a.) 22

b.) 23

c.) 24

d.) 25

~~P₁ P₂ P₃~~

~~5 8 13~~

+ 1

25

Page - 102

f - 20

EP

6

Q23.)

$$P_1 \quad P_2 \quad P_3 \quad P_4$$

4 3 7 ~~8~~

$$3 + 2 + 6 + n = 20$$

$$11 + n = 20$$

$$n = 20 - 11$$

$$n = 9.$$

$$3+2+6+8+1 \\ (20)$$

1.) Necessary Condition:—The deadlock may be possible or may not be possible.

2.) Sufficient Condition:—The deadlock never be possible.

→ (The least upper bound which satisfies the condition is called as sufficient condition).

Page -101)

Q13.)

$n \rightarrow$ processes

$m \rightarrow$ resources.

$p_i \rightarrow S_i$.

$\xrightarrow{?}$ individual

$$m=10$$

a.) $\forall i, S_i \leq m$

$$p_1 = 9$$

$$p_2 = 9$$

$$p_3 = 9$$

⋮

$$p_{100} = 9$$

b.) $\forall i, S_i \leq n$

✗

c.) $\sum_{i=1}^n S_i \leq m+n$

d.) $\sum_{i=1}^n S_i \leq (m \times n)$

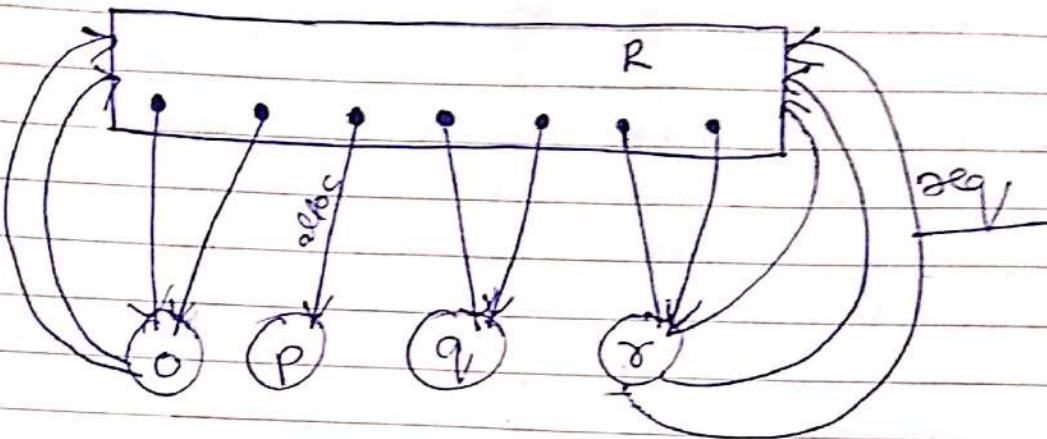
$$\sum_{i=1}^n S_i - n + 1 = m$$

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

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

Page - 100

(Q10.)



$$n = 4 (O, P, Q, R)$$

$$x_O = 2$$

$$x_P = 1$$

$$x_Q = 2$$

$$x_R = 2$$

$$y_O = 2$$

$$y_P = 0$$

$$y_Q = 0$$

$$y_R = 3$$

$$\boxed{y_P = y_Q = 0}$$

✓ (6 Ans.) $x_P + x_Q \geq \max y_k, k \in P, Q.$

available
resources

Concepts of Deadlocks :-

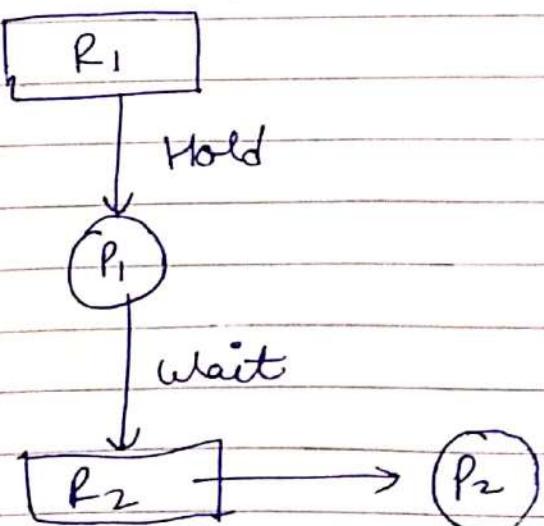
- 1.) Deadlock characteristics
- 2.) Deadlock prevention
- 3.) Deadlock avoidance
- 4.) Deadlock detection
- 5.) Deadlock recovery

Deadlock Characteristics :-

1.) Mutual Exclusion :→ The resource has to be allocated to only one process or it is freely available.

→ There should be a one-to-one relationship between the resource and the process.

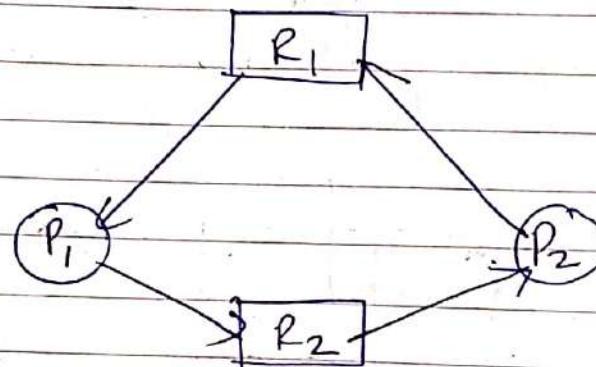
2.) Hold and wait :- The process is holding the resource and waiting on some other resource simultaneously.



3.) No Preemption:- The resource has to be "voluntarily" released by the process after completion of the execution.

→ 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.

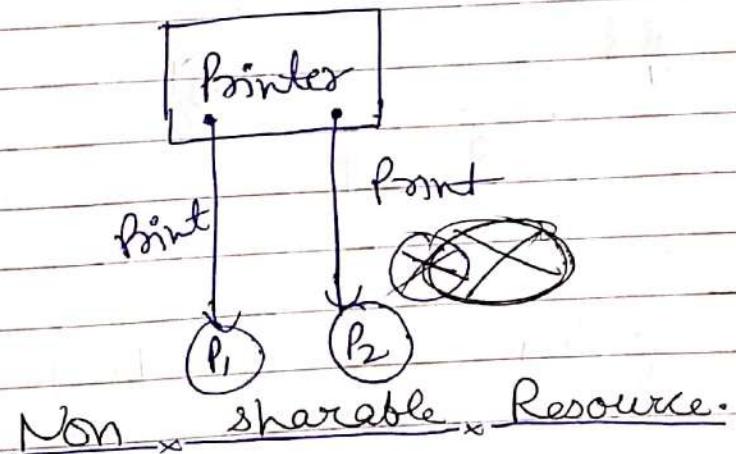
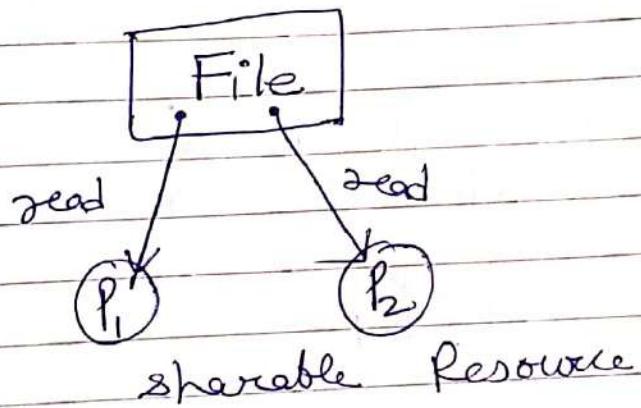


$R_1 \rightarrow P_1 \rightarrow R_2 \rightarrow P_2$

NOTE:- If all these above four conditions are existing simultaneously in the system, then definitely there exist a deadlock.

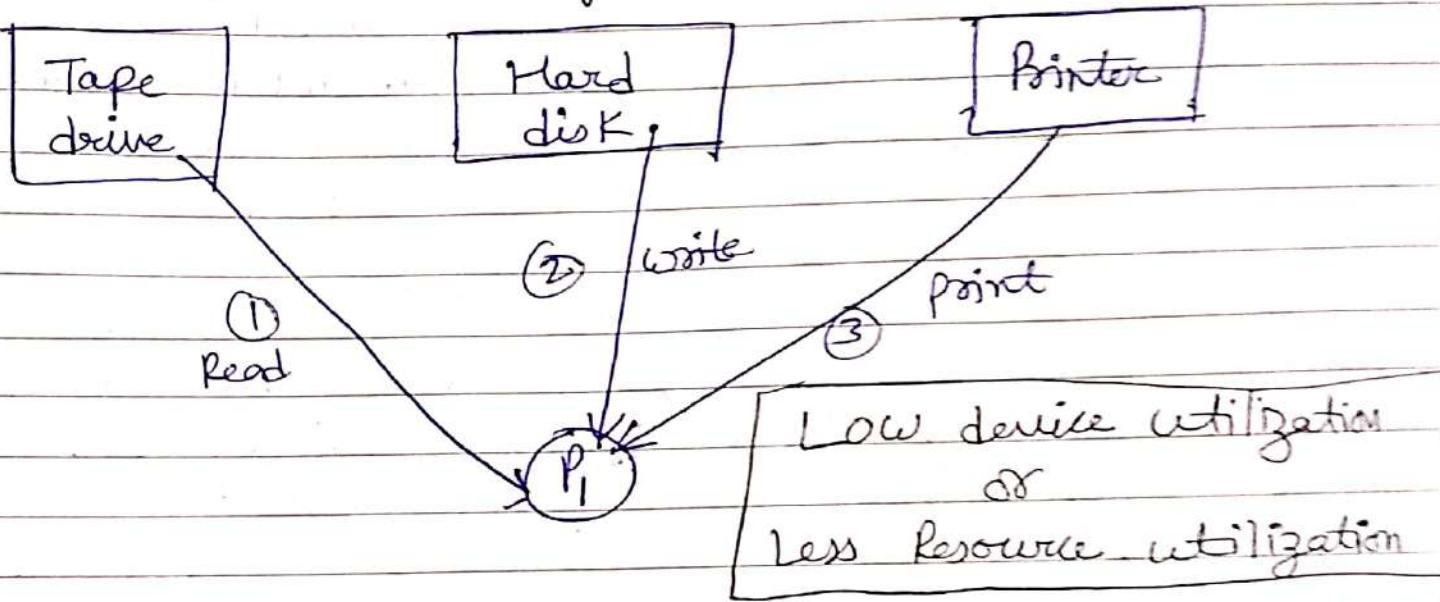
DEADLOCK PREVENTION:

1.) Mutual Exclusion:- It is not possible to disassociate the mutual exclusion always because of shareable and non shareable resources.

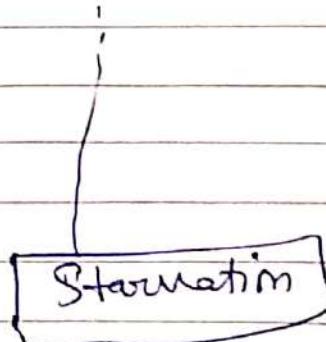
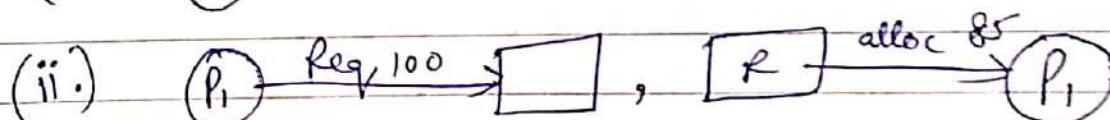
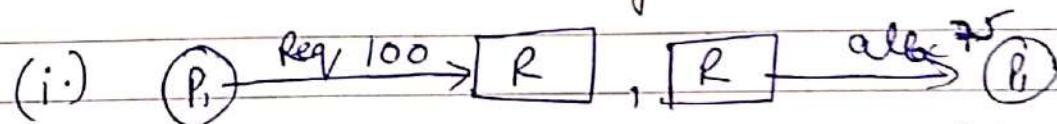


2) HOLD AND WAIT:-

(i) Allocate all the Resources required by the process before start of execution.



(ii.) The process should release all the existing resources, before making the new request.



NOTE:- If the above condition is followed to disatisfied hold & wait, then it may be possible that, the process will go into Starvation.

3.) NO PREEMPTION:- The process P_1 is requesting for resource ' R_1 '.

If the Resource R_1 is free then it will be allocated to the process ' P_1 '.

The Resource R_1 is not free, ~~then~~ and it is allocated to some other process ' P_2 '.

If the process ' P_2 ' is in the execution then process ' P_1 ' has to wait.

The process P_2 is not in the execution and it is waiting on some other resource ' R_2 '.

→ Preemption.
Preempt the resource R_1 from process P_2 and allocate it to the process ' P_1 '.

4.) CIRCULAR WAIT:-

- The resources will be assigned with a unique numerical numbers.
- The process can request for the resource only if + in the increasing (or decreasing) order of enumeration (numbering).

(increasing order)

$$1.) P_1 \rightarrow R_7$$



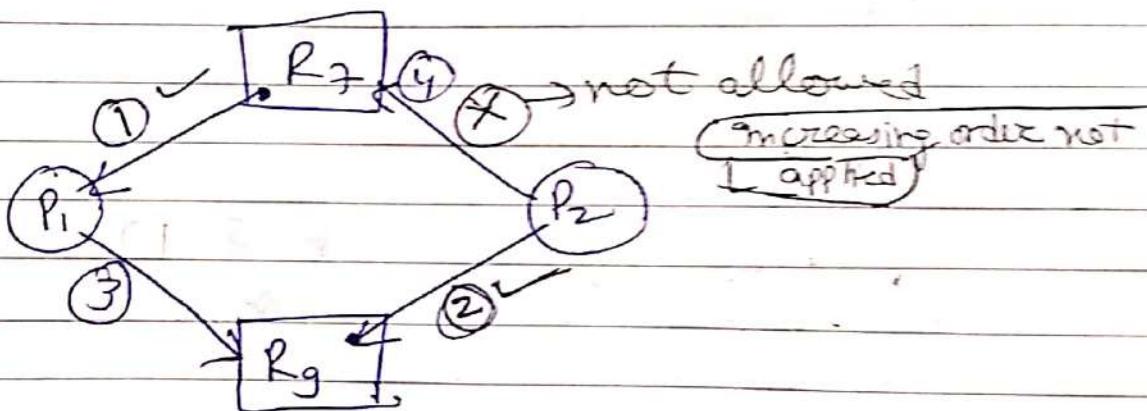
$$P_1 \rightarrow R_5$$



$$P_1 \rightarrow R_3$$



2.)



DEADLOCK AVOIDANCE :-

Deadlock avoidance is implemented by using Banker's algorithm:-

Banker's Algorithm :-

Processes $\rightarrow P_0, P_1, P_2, P_3, P_4$

Resources $\rightarrow A, B, C$.

Max Need			Current Allocation			Current available			Remaining Need				
A	B	C	A	B	C	A	B	C	A	B	C		
P ₀ \rightarrow	7	5	3	0	1	0	X	3	3	2	7	4	3 $\rightarrow P_0$
P ₁ \rightarrow	3	2	2	2	0	0	X	5	3	2	1	2	2 $\rightarrow P_1$
P ₂ \rightarrow	9	0	2	3	0	2	X	7	4	3	6	0	0 $\rightarrow P_2$
P ₃ \rightarrow	2	2	2	2	1	1	X	7	5	3	0	1	1 $\rightarrow P_3$
P ₄ \rightarrow	4	3	3	0	0	2	X	5	5	5	4	3	1 $\rightarrow P_4$
	<u>7 2 5</u>			<u>10 5 7</u>									

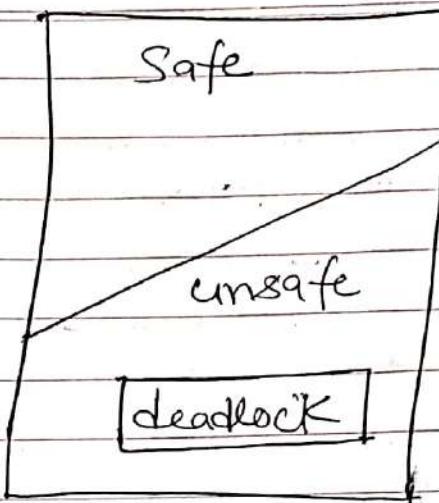
Total Available

A	B	C
\downarrow	\downarrow	\downarrow
10	5	7

Remaining need = Max need - current allocation

Current available = total Available - total Allocation

Safe Sequence :- P₁, P₃, P₀, P₂, P₄



→ If we can satisfy the remaining need of all the processes with the current available processes then system is said to be in the safe state otherwise system is said to be in the unsafe state.

→ If the system is in the unsafe state, then it is possible for the deadlock.

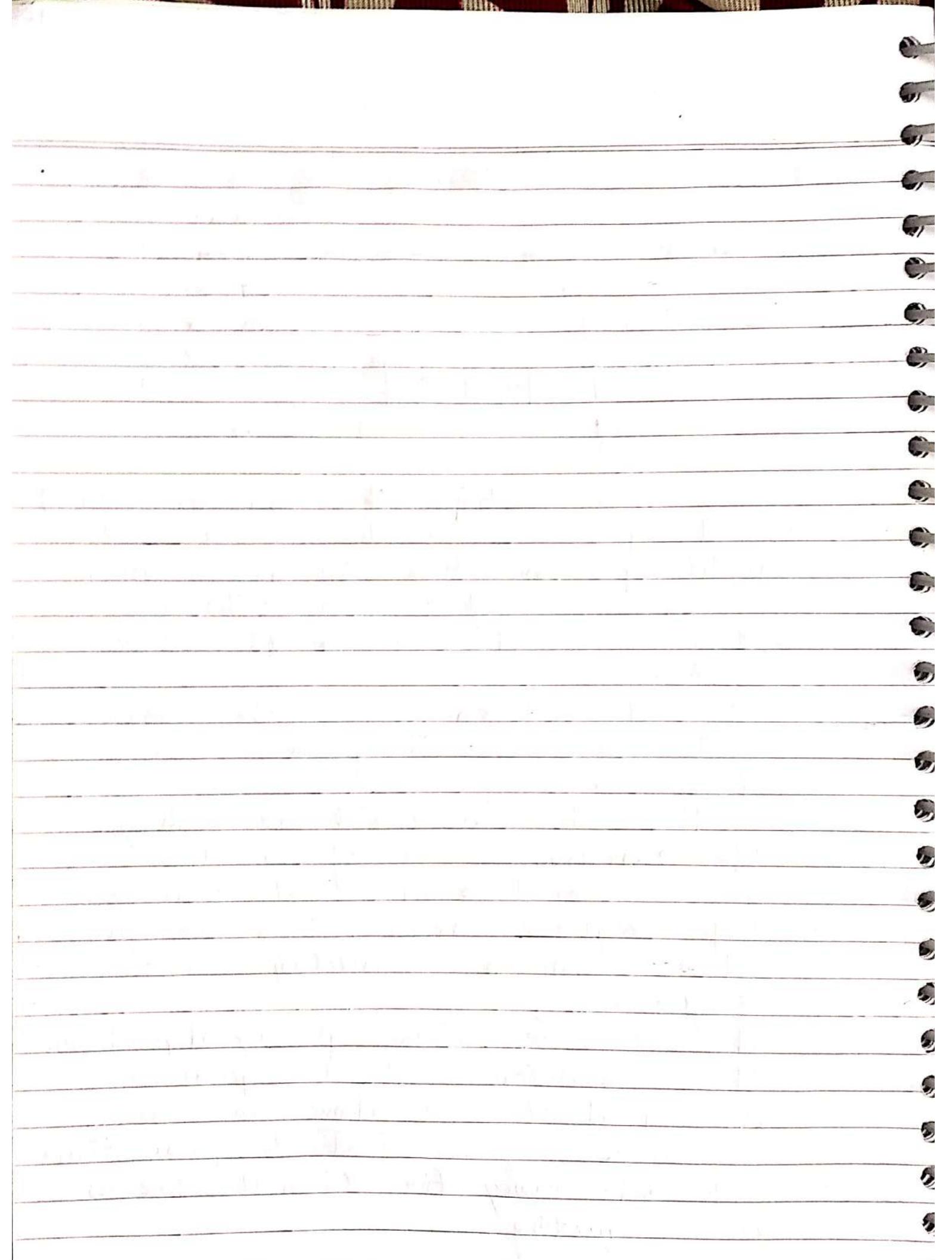
→ The order in which we satisfy the remaining need of all the processes is called safe sequence.

→ Safe sequence may not be unique, there can be multiple safe sequence.

→ The unsafe state purely depends on the behaviour of the processes.

→ The deadlock avoidance is less restrictive than deadlock prevention.

→ Each and every time when the process is requesting



P-99CrashQ2.)

4 6 3

Remaining

Need
A: B C

6 2 0

1 3 0

1 0 2

2 0 4

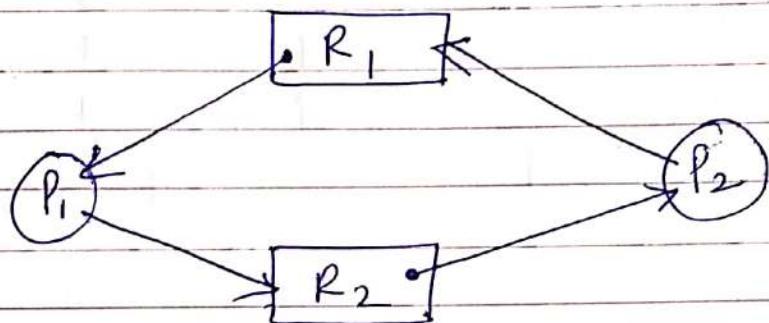
Process	Max Need	Current Allocation	Current Available			Remaining Need A: B C
			A	B	C	
P ₀ →	6 5 4	0 3 4	4	3	1	6 2 0
P ₁ →	3 4 2	2 1 2	6	4	3	1 3 0
P ₂ →	1 0 4	0 0 2	6	7	7	1 0 2
P ₃ →	3 2 5	1 2 1				2 0 4
		3 6 9				

P: P₁, P₀, P₂, P₃.Q: P₁, P₂, P₃, P₀

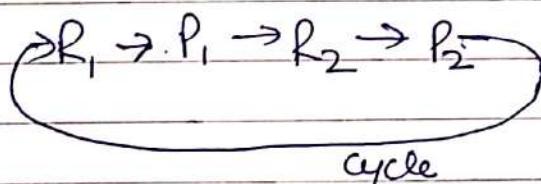
(a) P & Q only

DEADLOCK DETECTION :-

1.) The Resources are of single Instance Type :-



Rem: Cycle detection algorithm.



NOTE :- If all the resources are of single instance type, then the cycle in the resource allocation graph is necessary and sufficient condition for occurring of deadlock.

→ If all the resources are of non single instance type then cycle in the resource allocation graph is just a necessary condition but not sufficient condition for occurring of a deadlock.

(ii) The resource 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 ₀
P ₁	→ 2	0	0				2	0	2 → P ₁
P ₂	→ 3	0	3				0	0	0 → P ₂
P ₃	→ 2	1	1				1	0	0 → P ₃
P ₄	→ 0	0	2				0	0	2 → P ₄

Safe sequence :- P₀,

What happens if the process P₂ is requesting for additional resources of P₂ → (0, 0, 1).

$$\begin{array}{r}
 \text{000 (old)} \\
 + \text{001 (New)} \\
 \hline
 \text{001 (Latest)}
 \end{array}$$

Current allocation	Current available	Remaining Need
A B C	A B C	A B C
0 1 0	0 0 0	0 0 0 $\rightarrow P_0$
2 0 0	0 1 0	2 0 2 $\rightarrow P_1$
3 0 3		0 0 1 $\rightarrow P_2$
2 1 1		1 0 0 $\rightarrow P_3$
0 0 2		0 0 2 $\rightarrow P_4$

Safe Sequence :- $P_0 \Rightarrow$ unsafe \Rightarrow deadlock
 P_1, P_2, P_3, P_4

DEADLOCK RECOVERY:-

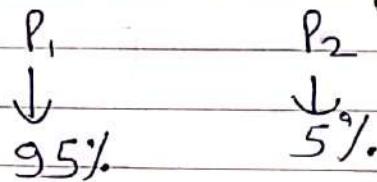
I) KILLING THE PROCESS:-

a.) Kill all the processes which are involved in the deadlock.

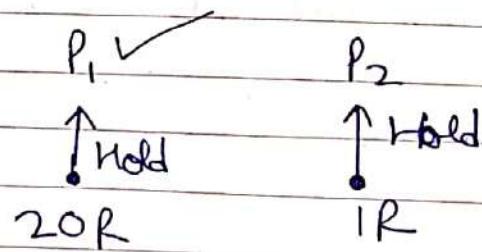
b.) Kill one after the other.

→ Low priority

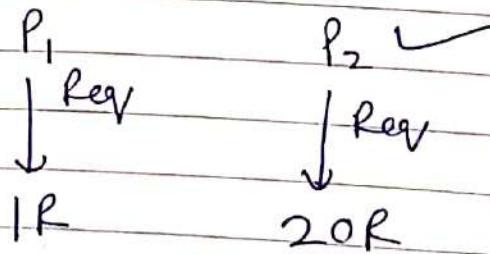
→ % of process completion



→ No. of Resources the process is Holding



→ No. of Resources the process Requesting.



2.) Resource Preemption:-

- Resources will be preempted from the processes which are in the deadlock and preempted resources will be allocated to some other processes, so that there is a possibility of recovering the system from the deadlock.
- If the above condition is followed to recover the system from deadlock, then there may be a possibility of the process to enter starvation.

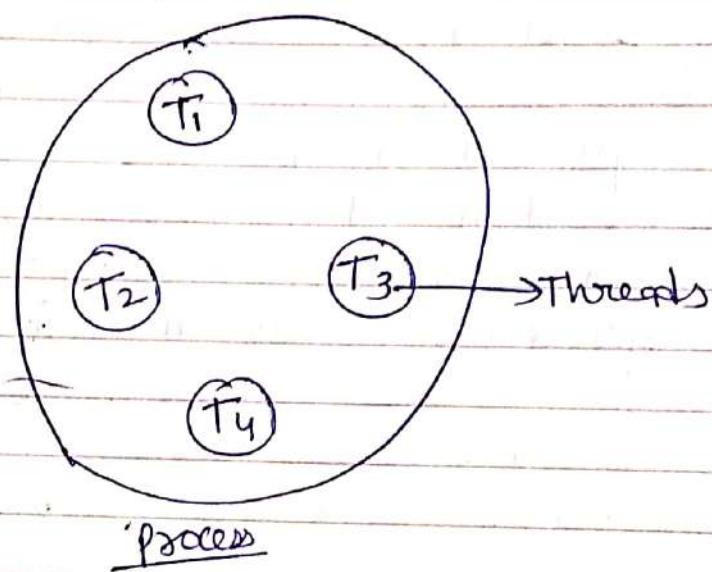
3.) Ostroich Algorithm:-

- ignore the deadlock.

THREADS:-

Def:- The light weight process

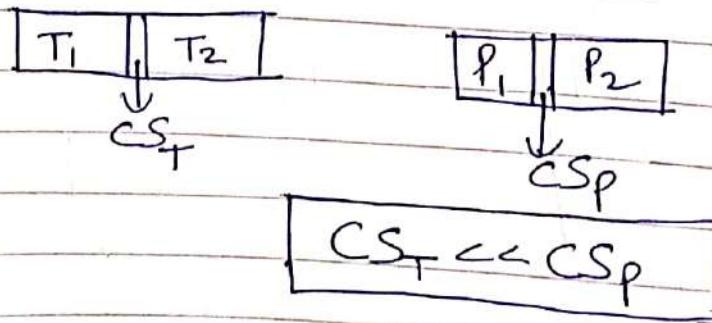
- No. of instructions will be less.
- context (Attributes) will be less.



Advantages of Threads:-

1) Responsiveness: If the process is divided into multiple threads, then if one thread is completed the execution immediately the output will be responded, this response will be faster compare to the response of the process.

2) Faster Context Switches:-

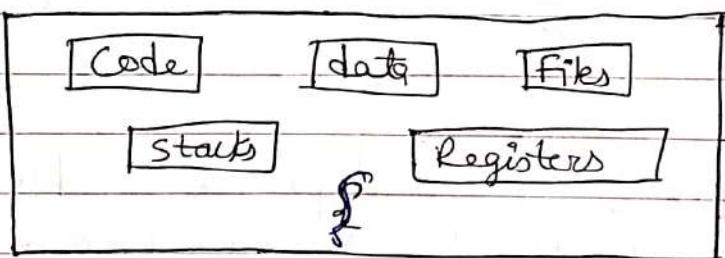


→ The context switching time between the threads will be very less compare to the context switching time b/w the processes because threads will have less context compare to the processes.

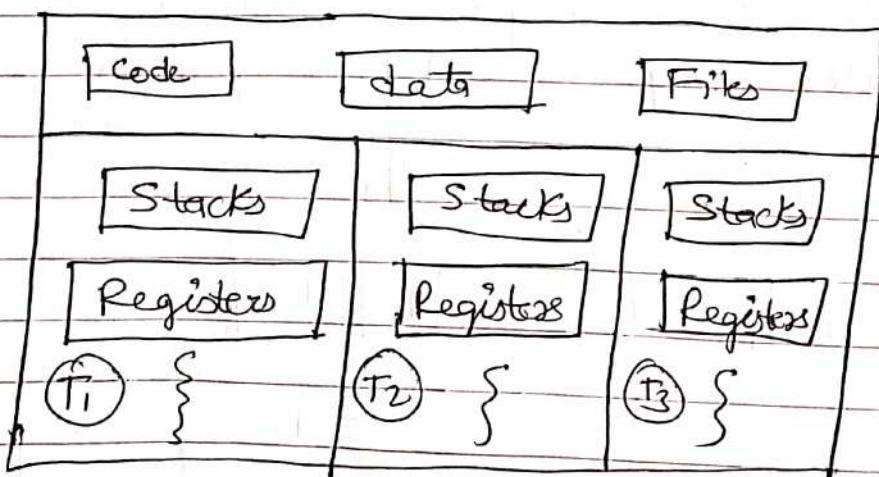
3.) Effective Utilization of Multi processor Systems:-

If the process is divided into multiple threads then different threads can be scheduled on to different processes, so that process execution will be faster.

4.) Resource Sharing:



Single threaded process



Multi threaded process

→ The Resources like code, data, (Global variables), files, heap and memory will be shared among all the threads within the process, but stacks and registers cannot be shared and every thread will have its own stacks and Registers.

5.) Enhanced Throughput of the System:-

→ If the process is divided into multiple threads and if we consider as one thread as one job, then No. of jobs completed per unit time will increase and throughput of the system will be enhanced.

6.) ECONOMICAL:-

→ The implementation of the threads does not require any cost and there are various programming language API which support Implementation of the threads.

Eg:- JAVA API

The threads are further categorized into two types:-

- 1.) User level threads.
- 2.) Kernel level threads.

User Level Threads

- 1.) User level threads are implemented by user or programmer.
- 2.) O.S. does not know about user level threads, it views User level thread as a process only.
It means O.S. can not recognize user level threads.
- 3.) If one user level thread is performing blocking system call, then the entire process will be blocked.
- 4.) User Level Threads are designed as dependent threads.

Kernel Level Threads

- 1.) Kernel level threads are implemented by OS.
- 2.) Kernel Level Threads are recognized by operating system.
- 3.) If one kernel level thread is performing blocking system call, then another thread will continue the execution.
- 4.) Kernel Level Threads are designed as independent threads.

5.) Implementation of user level thread is easy or simple.

5.) Implementation of Kernel level threads is complicated.

6.) User Level Threads will have less context.

6.) Kernel Level Threads will have more context.

7.) No Hardware support required for the User Level Threads.

7.) Scheduling of Kernel Level Threads requires Hardware Support.

NOTE:- "GANG SCHEDULING" is used in the Threads.

Workbook

P-79

Q14.) (d. Ans.)

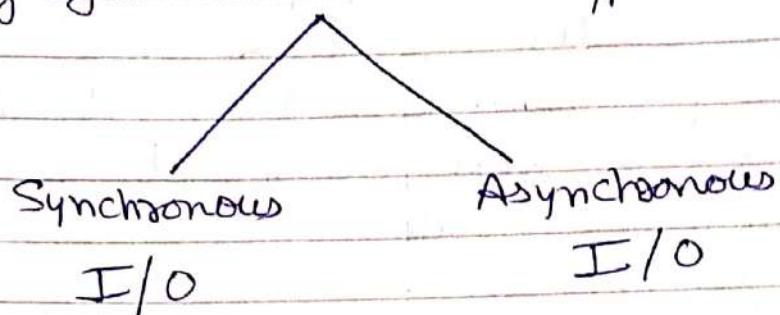
P-80

Q18.) (a) only i, ii & iii.

Q20.) (d) both heap & global variable.

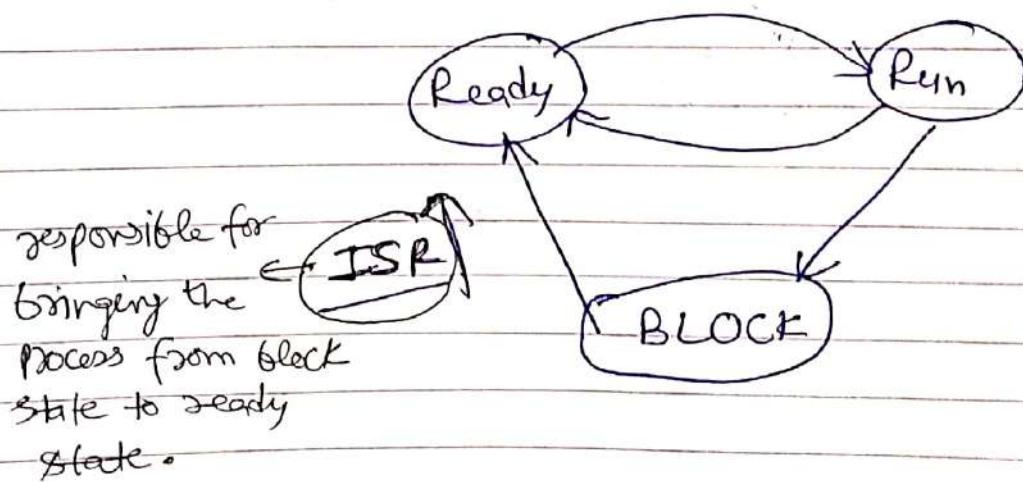
Q16.)

I/O Operation :- I/O of the process is categorized into two types

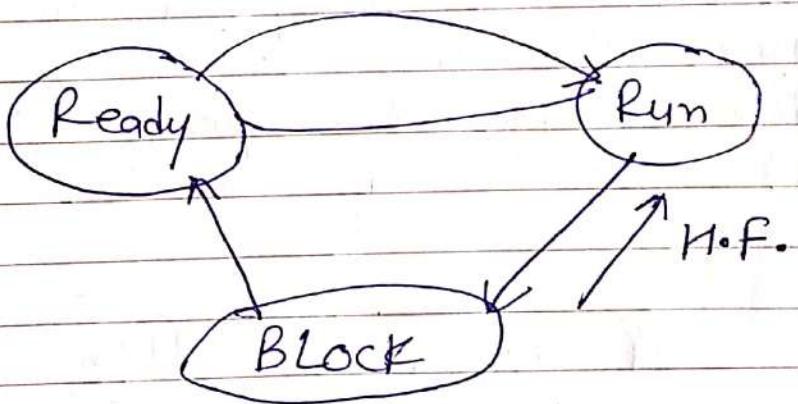


1.) SYNCHRONOUS I/O :- In Synchronous I/O, the process performing I/O operation will be placed in the blocked state till the I/O operation is completed.

→ Once I/O is completed, an interrupt Service Routine will be initiated which brings the process from block state to ready state.



- 2) ASYNCHRONOUS I/O :- In asynchronous I/O while initiating the I/O request, the "handler function" will be registered. The process is Not placed in the blocked state and it continues to execute the remaining code after initiating the I/O request.
- Once I/O is completed, the signal mechanism is used to notify the process that the data is available and "Register handler function" will be asynchronously invoked.



O.S. by

William Stallings

MEMORY MANAGEMENT:-

Main memory
 physical memory
 primary memory
 RAM memory

FUNCTIONALITY:- Allocating and deallocated
 the memory to the process.

GOAL:- The efficient Utilization of memory
 by minimizing the internal and external
 fragmentation.

Memory

→ Byte Addressable.
 Eg: 512 KB

→ Word Addressable.
 Eg: 512 KW

$$2^2 = 4 \text{ bytes.}$$

$$2^3 = 8$$

$$2^4 = 16$$

$$2^5 = 32$$

$$2^6 = 64$$

$$2^7 = 128$$

$$2^8 = 256$$

$$2^9 = 512$$

$$2^{10} = 1024 \approx 10^3 = 1K$$

$$2^{20} = 1M$$

1 Byte = 8 bits
 1 word = ?

$$2^{40} = 1T.$$

(bit) ~~1~~

8 bit → 16 bit

		→ 16 bits
0	1	000
1	2	001
2	3	010
3	4	011
4	5	100
5	6	101
6	7	110
7	Memory	111

binary bit

$3 = 2^3 = 8$ words

Capacity of the memory = No. of words in memory × word size

$$= 8 \times 16 \text{ bits}$$

$$= 8 \times 2 \text{ bytes.}$$

$$= 16 \text{ bytes.}$$

Q.) Consider a system which has 256 Mega words and each word is having size of 128 bits then what is the capacity of memory in bytes

$$= 128 \times 256 \text{ Megabytes}$$

~~$= 128 \times 256 \text{ Megabytes}$~~

$$= 2^{10} \times 2^{20} \text{ bytes}$$

~~2^{10}~~ =

$$= 2^{32} \text{ bytes} = 2^2 \times 2^{30} \text{ B}$$

$$= 4 \times 1 \text{ GB}$$

$$= 4 \text{ GB.}$$

$$\begin{array}{r} 32 \\ = \end{array}$$

$$\begin{array}{r} 264 \\ \times 6 \\ \hline 384 \end{array}$$

Q.) Consider a system which have 64 words & each one having size of 48 bits then what is the capacity of Memory in bytes?

$$\Rightarrow 2^{30} \times 2^6 \times 6$$

$$\begin{array}{r} 2^{30} \\ \times 6 \\ \hline 384 \end{array}$$

$$= [384 \text{ GB.}]$$

Q.) Consider a system where 32 binary bits are used to represent all the words of the memory. And each word is having the size of 32 bits. What is the capacity of memory in bytes?

$$= 4 \times 2^{32}$$

$$= 4 \times 2^{32}$$

$$= 4 \times 2^{30} \times 2^2$$

$$= [4 \times 1024 \text{ GB}]$$

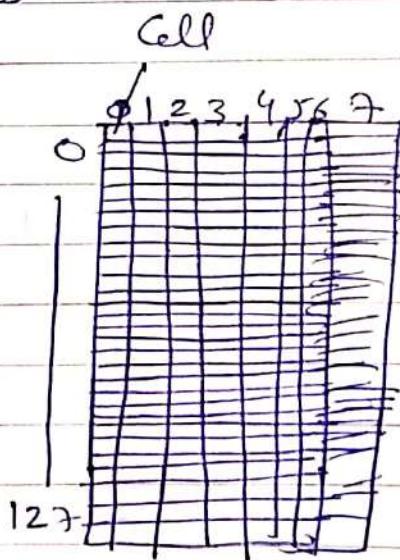
$$\begin{array}{r} \cancel{21} \\ \times \cancel{8} \\ \hline \cancel{64} \end{array}$$

RAM CHIP IMPLEMENTATION :-

RAM CHIP SIZE = 128 Byte x 16 KB
 organize the main memory capacity of 16 KB.

- 1.) No. of RAM chips required?
- 2.) Draw the memory organization? (byte)
- 3.) what is the size of Decoder required?
- 4.) Draw the memory organization map with the word size of 16 bits.

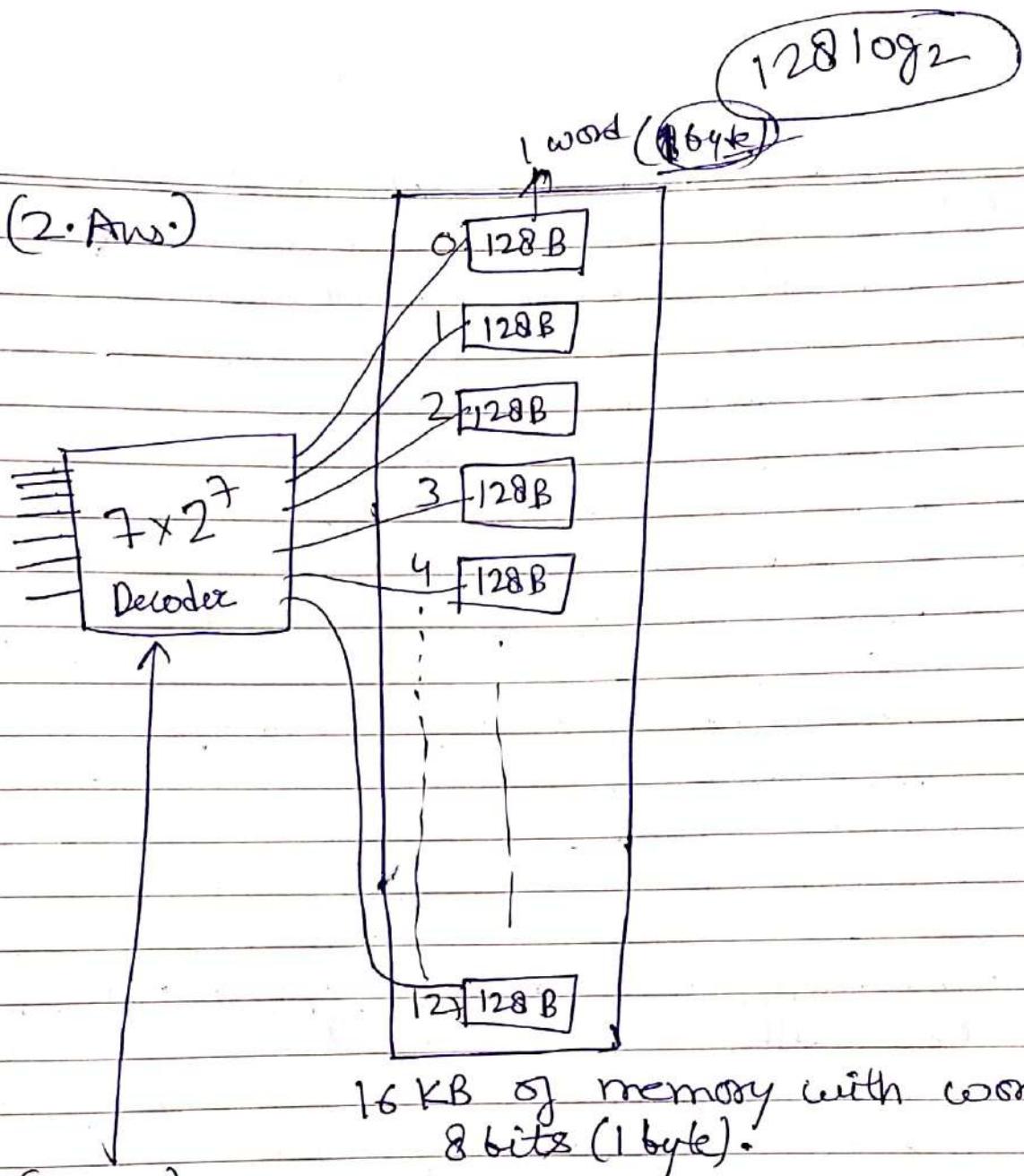
RAM CHIP SIZE = 128 B



Q1.) No of RAM chips required = ?

$$= \frac{16 \text{ KB}}{128 \text{ B}} = \frac{2^4 \times 2^{10}}{2^7} = \underline{\underline{2^7}} = 128.$$

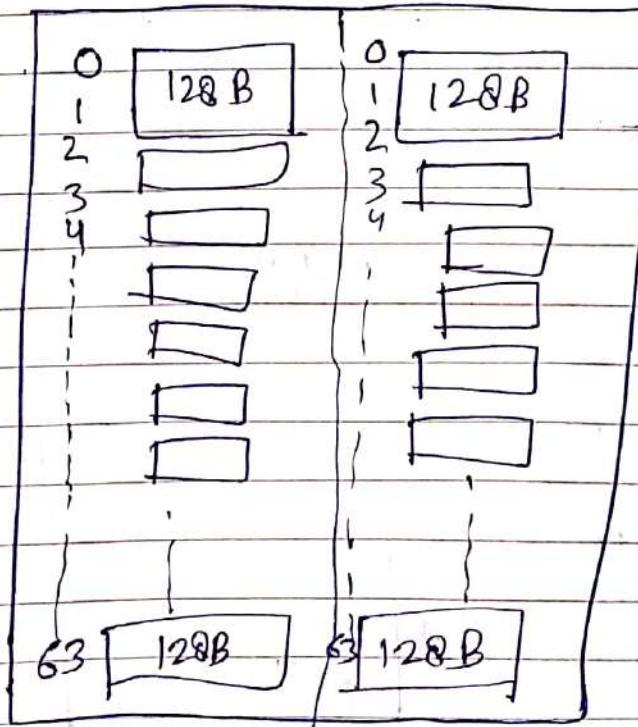
(2. Ans.)



16 KB of memory with word size as
8 bits (1 byte).

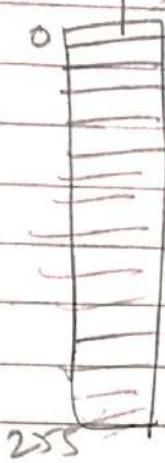
(3. Ans.)

(4. Ans.)



$$\frac{2^0}{2^5} = \frac{2^{15}}{2^5} = \frac{32\text{ byte}}{2^5} = 32$$

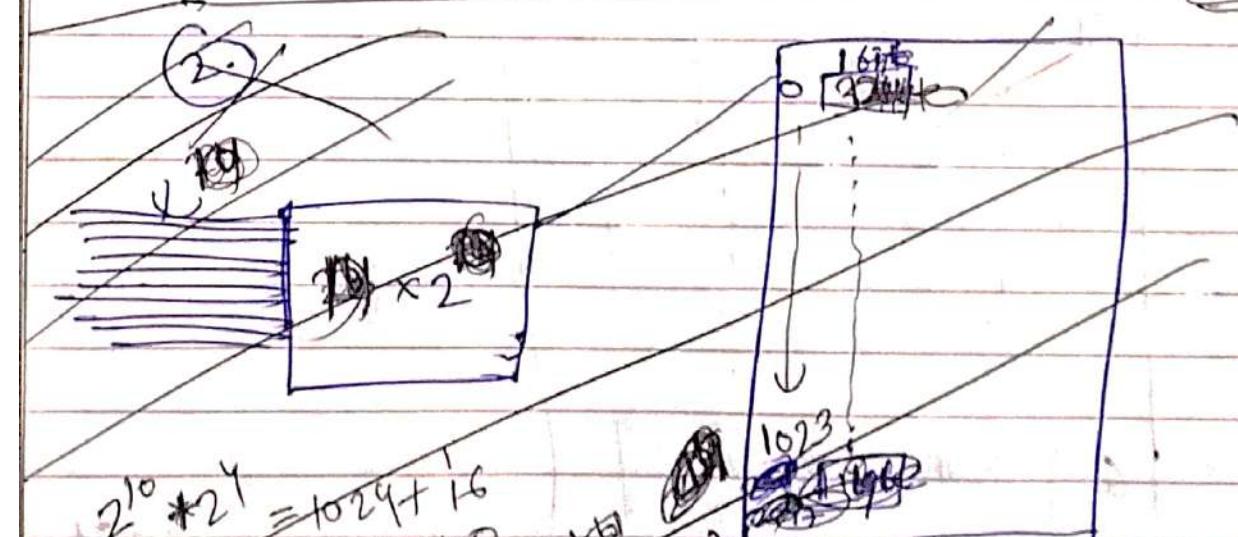
Q2.) RAM chip size = 256×1 -bit
 Organize the main memory capacity of
 $\Rightarrow 1$ bit 32 MB .



① RAM CHIP SIZE = $\frac{32\text{ MB}}{32\text{ B}}$

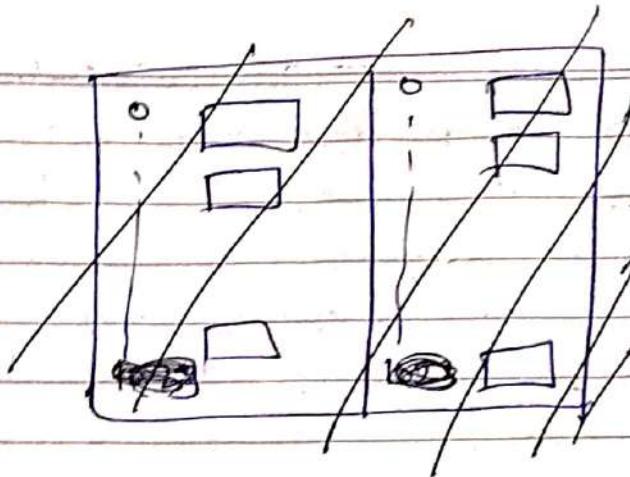
$$= \cancel{32} \cdot 2^{20} \times \cancel{32} \\ = 2^{20} \cdot \frac{32}{32}$$

$$2^{20} = \cancel{2^{20+1}} [1024]$$

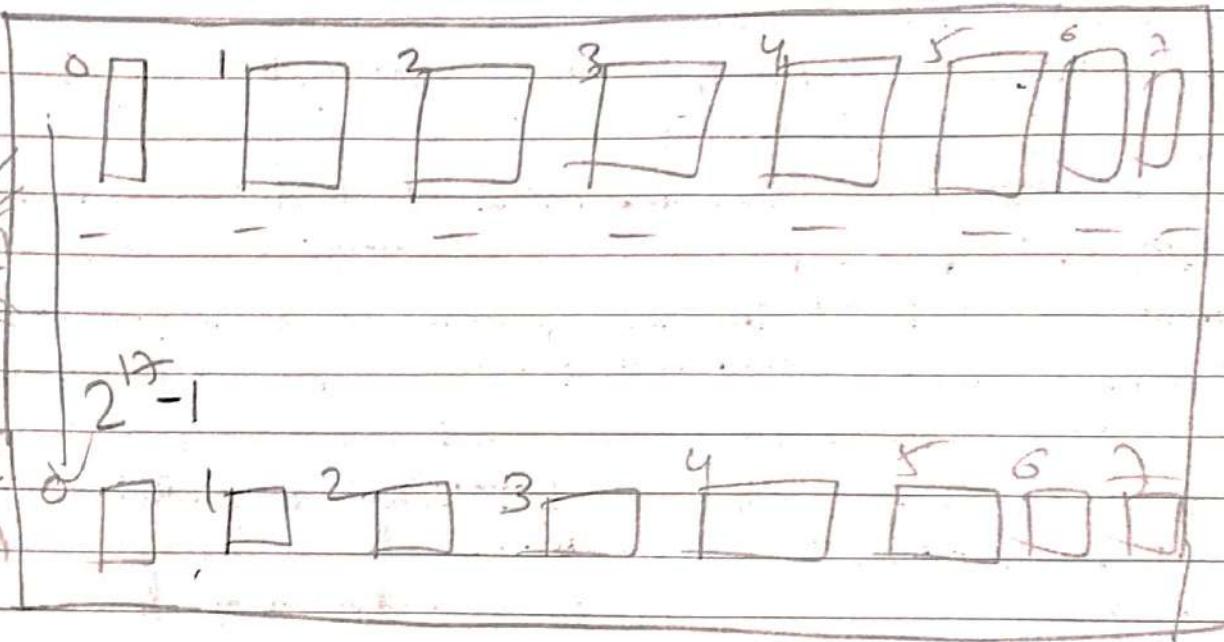


$$2^{10} \cdot 2^4 = 1024 + 16$$

4.)



2.)



$$= \frac{2^{20}}{8} = \frac{2^{20}}{2^3} = 2^{17} =$$

32 MB of memory with word size
8 bits.

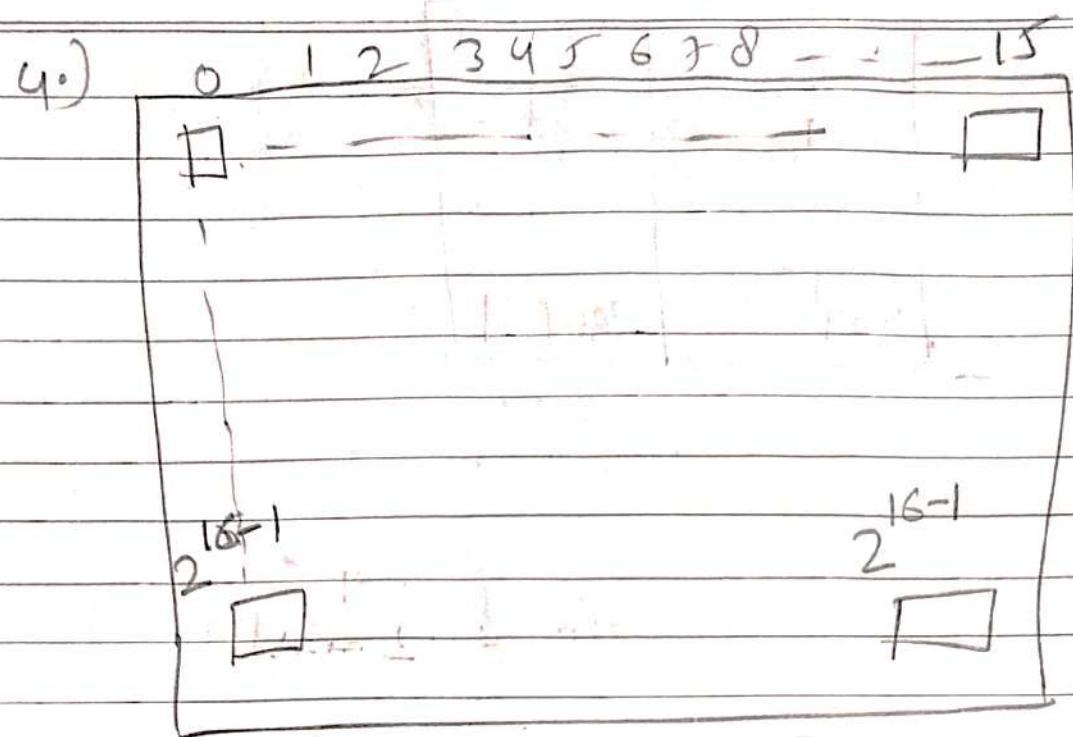
3.)

 17×2^1

Decoder

17

~~2²⁰~~ 2¹⁶



$$\cancel{2^{20}}_{16} = \cancel{2^{20}}_{2^4} = \cancel{2^{16}}$$

32 MB of memory with word size of 16 bits!

$$\frac{1024}{8} = 128$$

1024
8
128
64

Q.) RAM chip size = 512×2 -bits
 organize the main memory capacity of 16 MB.

$$\oplus \quad \begin{array}{c} 16 \text{ MB} \\ \hline 512 \times 2 \text{ bits} \end{array}$$

$$= \frac{16 \text{ MB}}{512 \times 2 \text{ bits}}$$

$$= \frac{16 \text{ M} \times 8 \text{ bits}}{512 \times 2 \text{ bits}}$$

$$= \frac{12^4 \times 2^{20} \times 2^3}{2^9 \times 2^1}$$

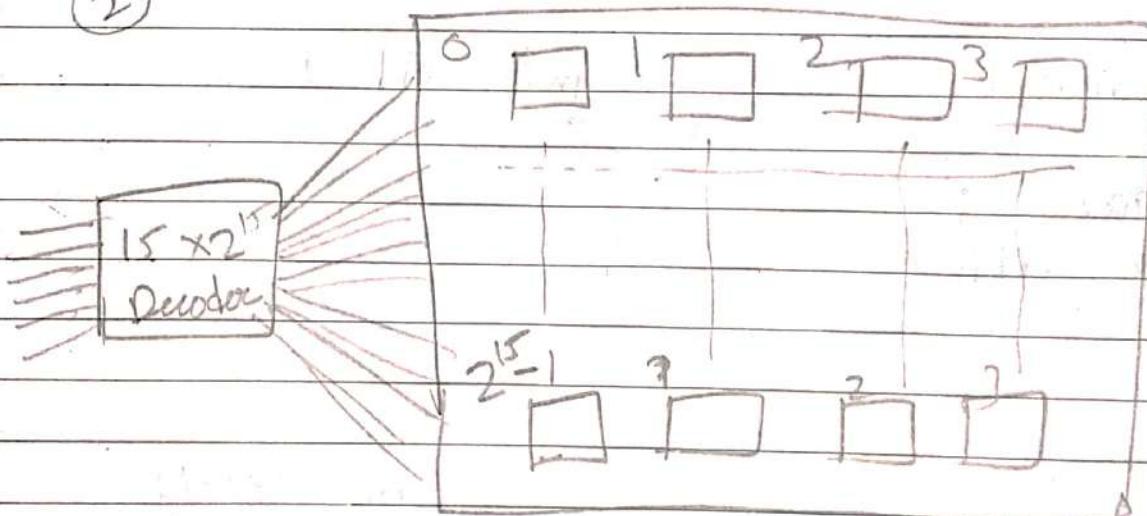
$$= \frac{2^2}{2^1}$$

$$= \frac{2^{10}}{2^1}$$

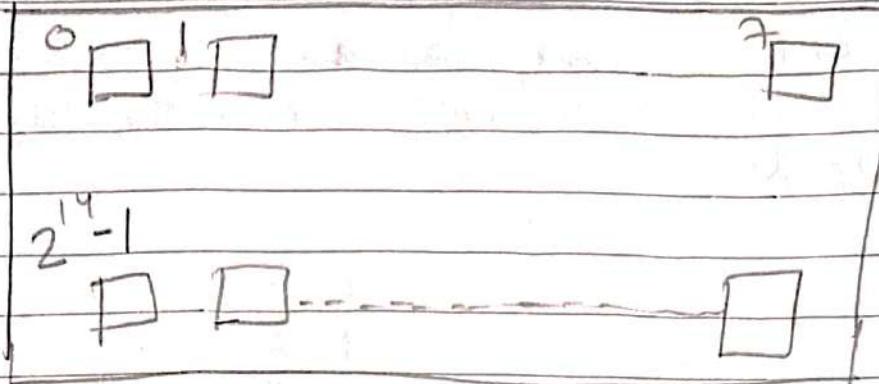
$$= 2^9$$

$$10 = 1128$$

(2)



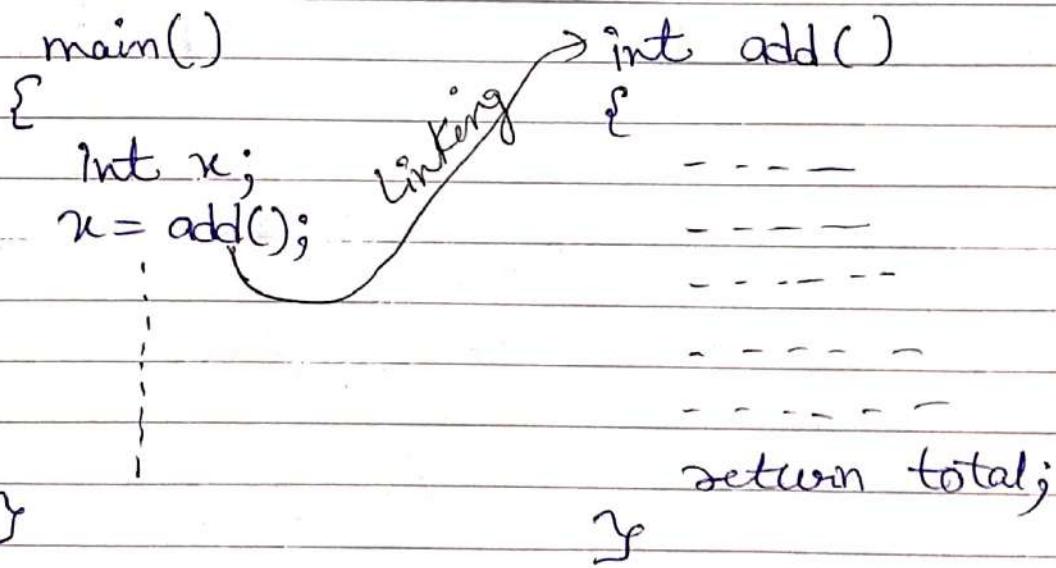
$$\frac{2^{17}}{2^3}$$



LOADING, LINKING AND ADDRESS BINDING :-

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

LINKING :- Establishing the linking between all the modules or all the functions of a program in order to continue the program execution is called as linking.



Loading and Linking is further categorized into two types:-

- 1.) Static
- 2.) Dynamic

1.) STATIC :- Loading the entire program into memory before start of program execution.

→ Inefficient utilization of memory, because whether it is required or not required, the ^{entire} program will be load into main memory.

→ Program execution will be faster.

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

2.) DYNAMIC :- Loading the program into memory on demand is called as Dynamic loading.

→ Efficient Utilization of a memory.

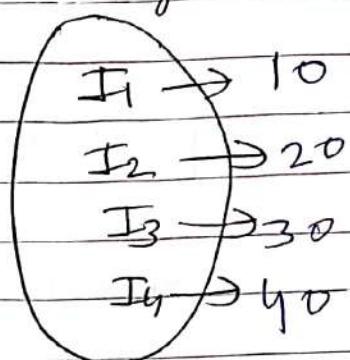
→ Program execution will be slower.

→ If the dynamic loading is used accordingly, dynamic linking will be applied.

ADDRESS BINDING :-



Program = p_1



PC = 10, 20, 30, 40

Memory

Association of program instructions and data to the actual physical memory locations is called as address binding.

→ Address Binding is categorized into 3 types

- Compile Time AB (Compiler)
- Load Time AB (Loader)
- Execution Time A.B or Dynamic AB.
(processor)

1.) Compile Time AB :- If the compiler is responsible of performing address binding, then it is called as compile time addressing binding.

→ This type of Address Binding will be done before loading the program into memory.

→ The compiler requires to interact with the O.S.

compile time Addressing binding.

2.) LOAD TIME AB :-

- This type of Address binding will be done after loading the program into memory.
- Load Time AB will be done by O.S. memory manager ("Loader")

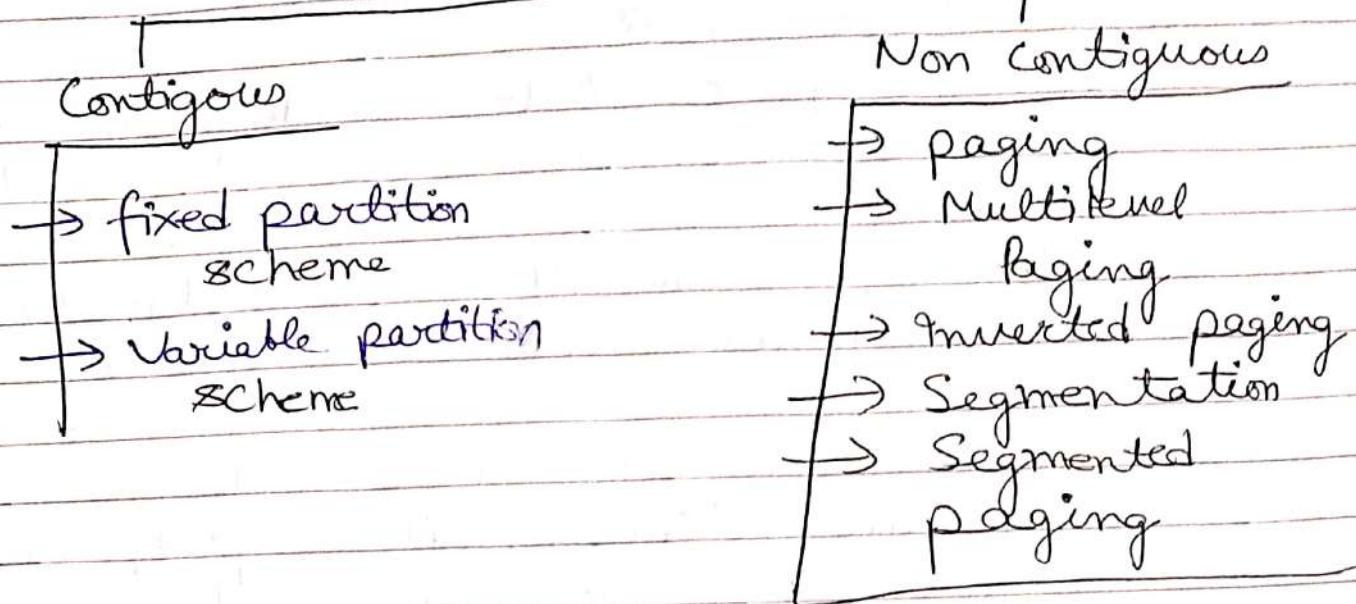
3.) EXECUTION TIME AB OR DYNAMIC AB :-

- The Address binding will be postponed even after loading the program into memory.
- The program will keep on changing the locations in the memory till the time of program execution.
- This type of Address Binding will be done by the processor at a time of program execution.

NOTE → Many of the ~~processes~~ operating system practically implement dynamic loading, dynamic linking and dynamic Address binding.

Ex:- Windows, Unix, Linux.

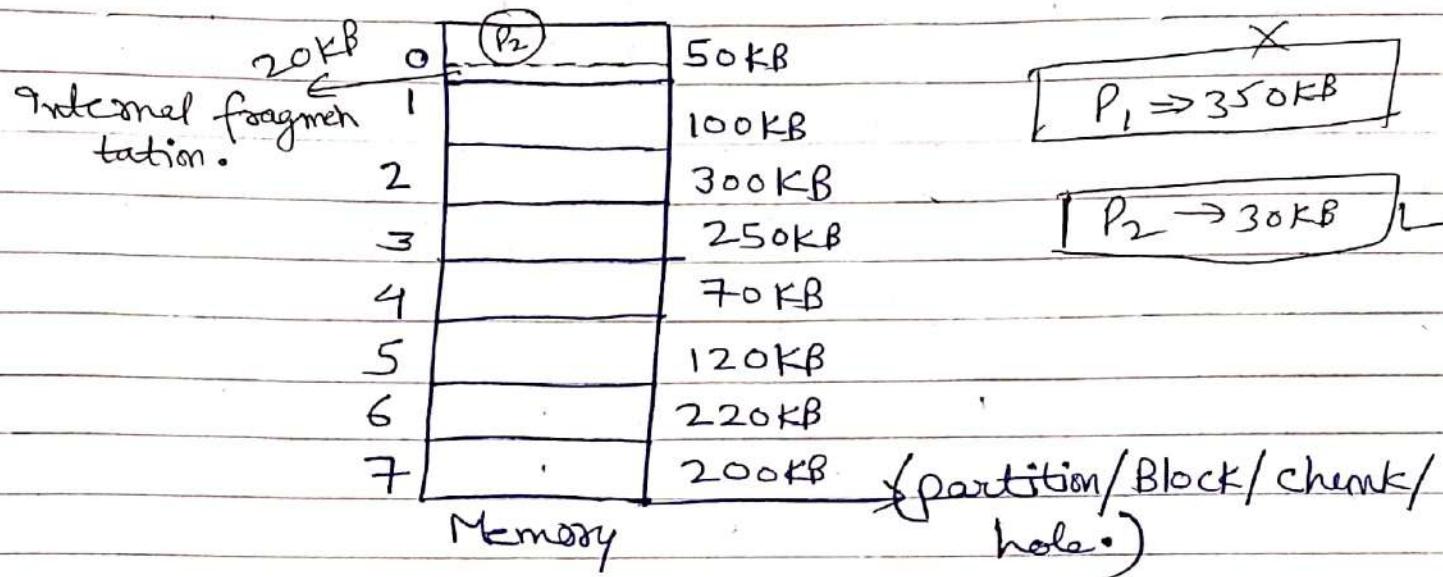
MEMORY MANAGEMENT TECHNIQUES:



Memory Management
Techniques

CONTINUOUS MEMORY MANAGEMENT:-

1) FIXED PARTITION SCHEME:-

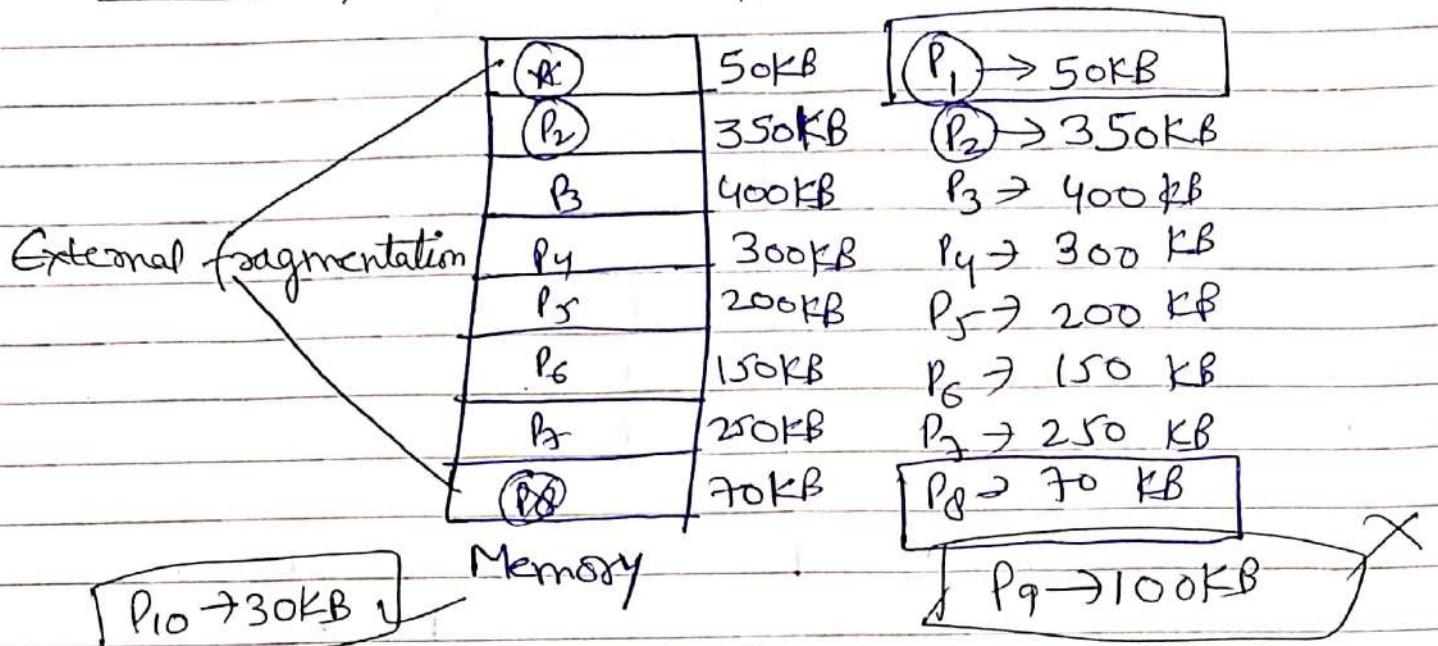


- In the fixed partition scheme, memory will be divided into fixed number of partitions.
- Fixed means no. of partitions are fixed in a memory.
- In every partition only one process will be accommodated.
- The degree of multiprogramming is restricted by no. of partitions in the Memory.
- Maximum size of a process is restricted by maximum size of a partition.
- Every partition is associated with the limit registers.

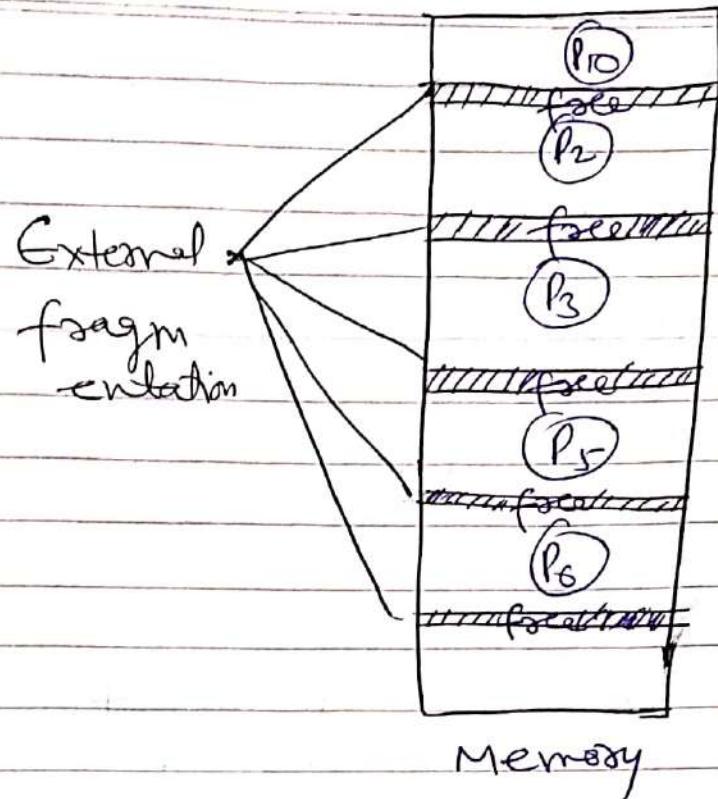
Limit Registers

- Lower Limit:- starting address of the partition
- Upper Limit:- ending Address of the position.

VARIABLE PARTITION SCHEME:-



- In the Variable Partition scheme, initially memory will be single continuous free block.
- Whenever the request by the process arrives accordingly partition will be made in the memory.
- If the smaller processes keep on coming, then the larger partition will ~~be made~~ be made into smaller partitions.



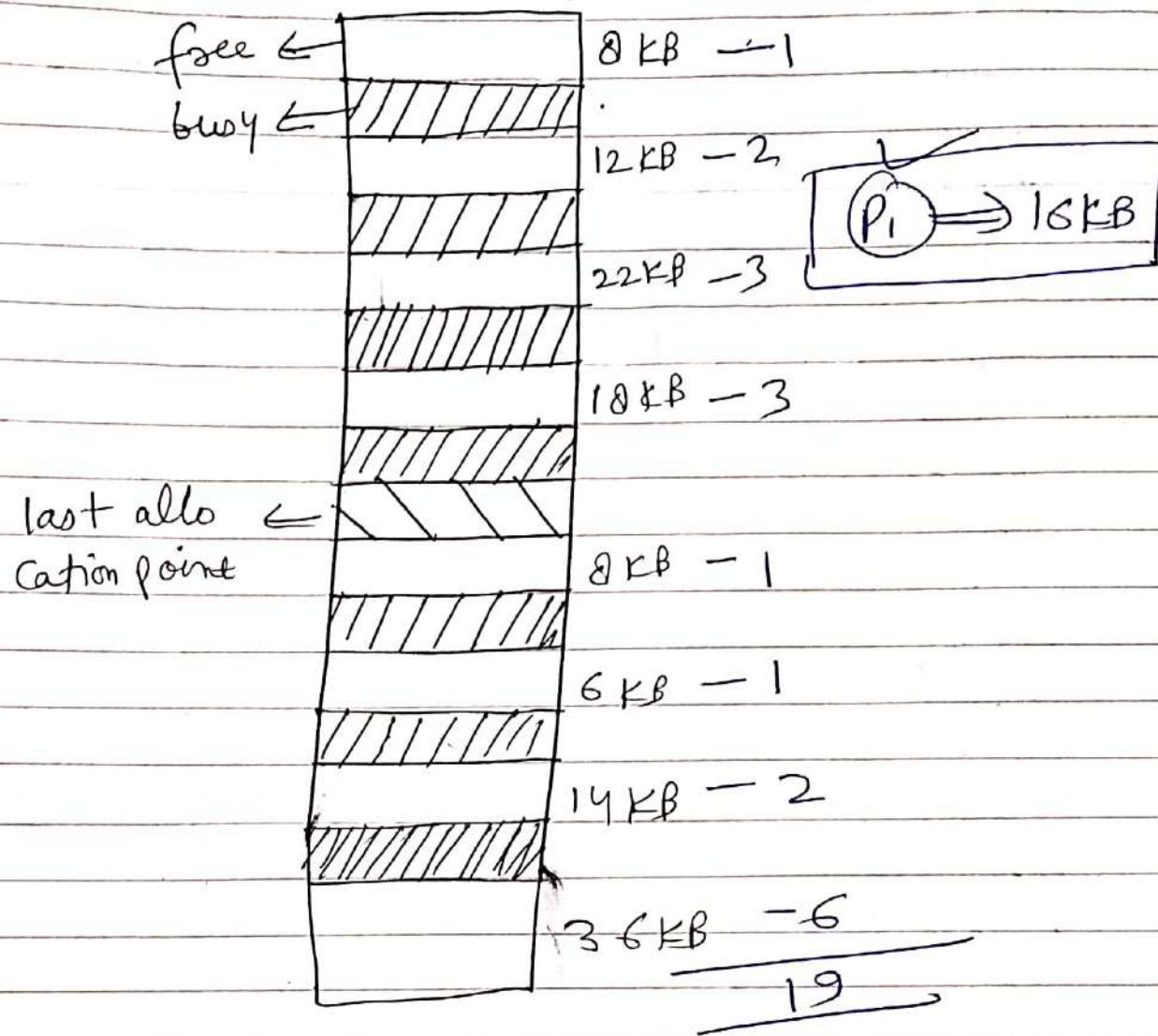
To avoid the problem of external fragmentation the following techniques are used.

- 1.) Compaction:- Moving all the processes toward the top or towards the bottom to make the free available memory in a single continuous place is called as compaction.
→ Compaction is undesirable to implement because it interrupts all the running processes in the memory.
- 2.) Implementing Non - Contiguous memory management techniques.

PARTITION ALLOCATION METHODS:-

- 1.) FIRST FIT:- Allocate the process in a partition which is first sufficient partition from the top of the memory.
- 2.) BEST FIT:- Allocate the process in a partition which is smallest sufficient among the free available partitions.
→ to find out the smallest sufficient, it requires to search all the ~~three~~ free partitions in the memory.
- 3.) WORST FIT:- Allocate the process in a partition, which is largest sufficient among the free available partitions.
→ to find out the largest sufficient, it requires to search all the ~~three~~ free partition in the memory.
- 4.) Next FIT:- The next fit also works like first fit, but it will search for the first sufficient partition from "last allocation point".

Q1.) Consider the following memory map,



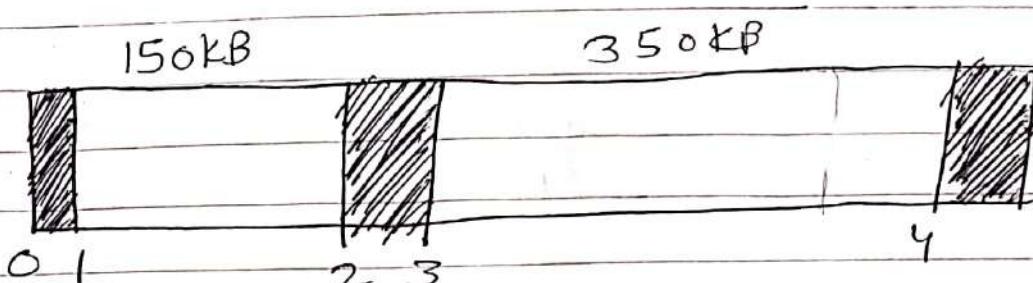
- 1.) First Fit: 22 KB ,
- 2.) Best Fit: 18 KB ,
- 3.) Worst Fit: 36 KB ,
- 4.) Next Fit: ~~14 KB~~ 36 KB ,

Q2.) How many successive requests of 6 KB will be satisfied by using First Fit assuming variable partition scheme?

- a.) 17 e.) 21
- b.) 18
- c.) 19

PF 50, 300

Consider the following memory map
Requests from the processes are 300KB,
25KB, 125KB 50KB respectively.



The above requests could be satisfied
with? (Assume variable partition
scheme.)

- a.) Best fit but Not first fit
- b.) First fit but Not Best fit
- c.) Both first fit and Best fit
- d.) None of the above.

(6 Ans.)

a) 100 X

X 300 50

PAGING

NON CONTIGUOUS MEMORY MANAGEMENT TECHNIQUES:-

PAGING:- It is using the terminology:-

1.) Logical Address space (L.A.S) or virtual address space (V.A.S)

⇒ Represented in the form of words or Bytes
e.g., 512KW or 512KB.

2.) Logical Address (L.A.) or Virtual Address (V.A.)

⇒ Represented in the form of bits.
e.g. 19 bits.

3.) Physical Address space (P.A.S.)

⇒ Represented in the form of words or Bytes.
Eg. 32MW or 32MB.

4.) Physical Address (P.A.)

⇒ Represented in the form of bits.
Eg :- 25 bits.

E.g ①: L.A = 39 bits

$$\begin{aligned} L.A.S &= 2^{39} W \\ &= 2^9 \times 2^{30} W \\ &= 512 GW. \end{aligned}$$

E.g. ② L.A.S = 256 MW

$$\begin{aligned} L.A &= 2^8 \times 2^{20} \\ &= 28 \text{ bits.} \end{aligned}$$

9/02/19

$$L \cdot A > P \cdot A$$

E.g. 3.) $P \cdot A = 45$ bits
 $\Rightarrow P \cdot A \cdot S = 2^{45}$ bits
 $= 2^5 \times 2^{40} B$
 $P \cdot A \cdot S = 32TB$

E.g. 4.) $P \cdot A \cdot S = 64KB$
 $\Rightarrow P \cdot A = 2^6 \times 2^{10}$
 $P \cdot A = 16$ bits.

→ The technique of mapping CPU generated logical address to physical address is called as Paging.

→ Paging is implemented in the hardware level.

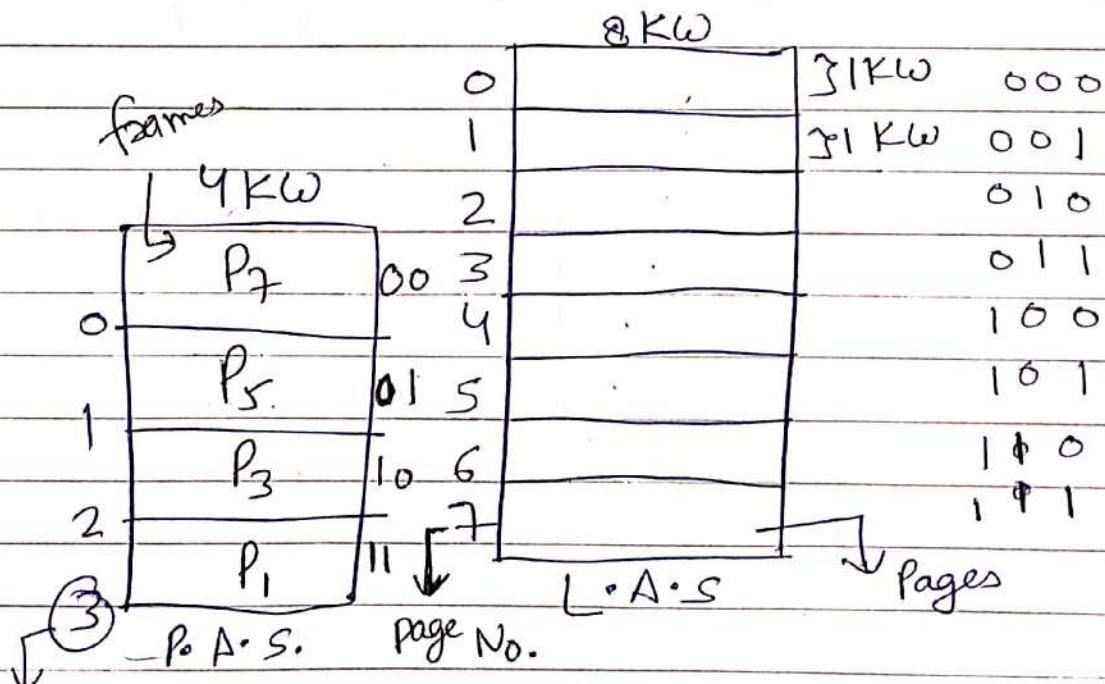
Let us assume

$$L \cdot A = 13$$
 bits

$$P \cdot A = 12$$
 bits

$$L \cdot A \cdot S = 2^{13}W = 8KW.$$

$$P \cdot A \cdot S = 2^{12}W = 4KW.$$



→ The logical address space will be divided into equal size pages.

$$\boxed{\text{No. of Pages} = \frac{\text{L.A.S.}}{\text{Page size}}}$$

$$= \frac{8\text{Kw}}{1\text{Kw}} = 8.$$

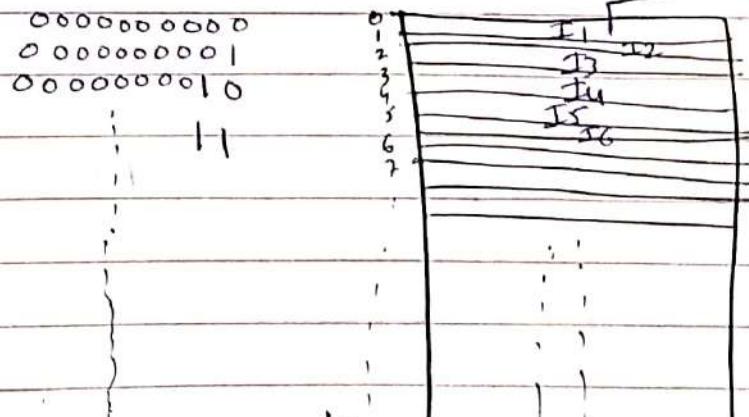
⇒ frames: The physical address space is divided into equal size frames.

$$\boxed{\text{Page Size} = \text{frame size}}$$

$$\boxed{\text{No. of frames} = \frac{\text{P.A.S.}}{\text{frame size}}} = \frac{4\text{Kw}}{1\text{Kw}} = 4.$$

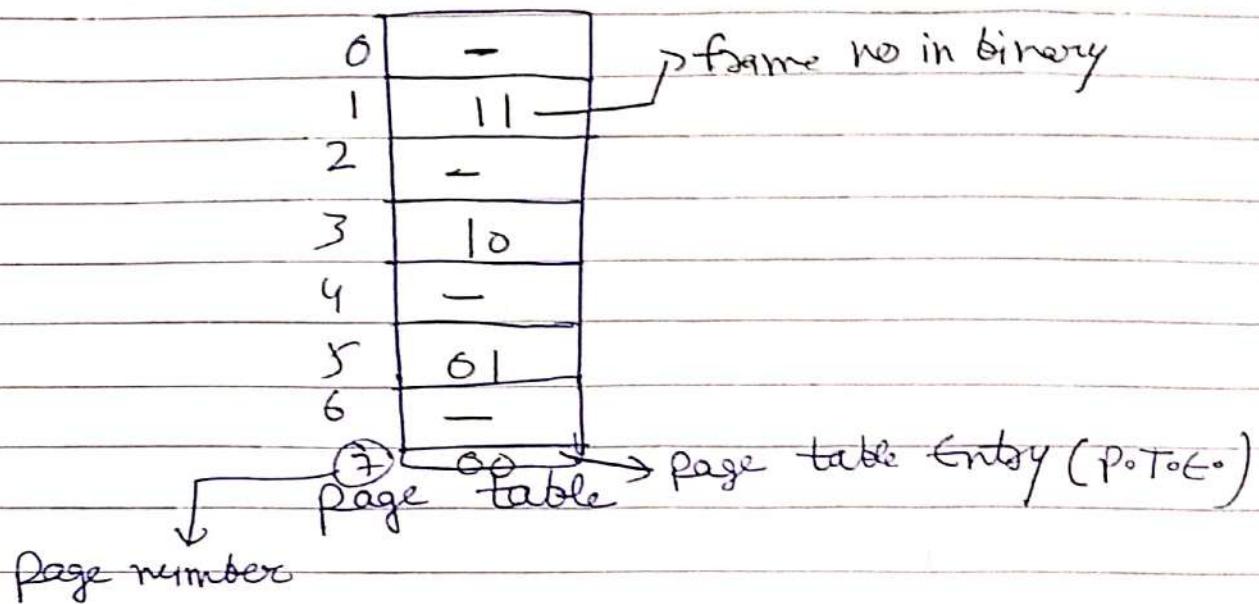
→ Some of the pages of L.A.S. will be brought into P.A.S.

$$\boxed{\text{page size} = 1\text{Kw}}$$



actual instruction of
the process.

→ Whenever the paging is applied, the page table has to be maintained.

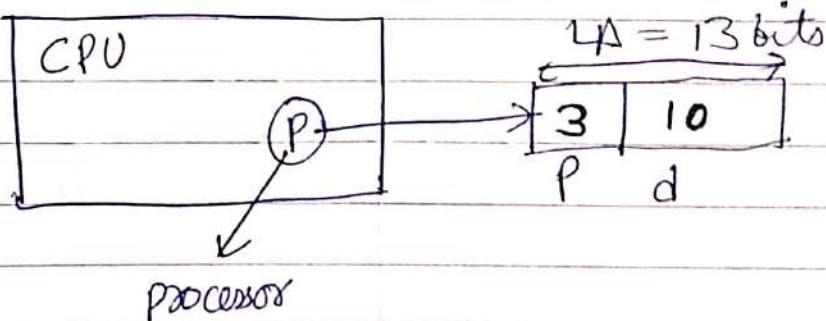


→ No. of entries in the page table is same as no of pages in L.A.S..

→ Page table entry definitely contains frame number.

→ Page table is also called as address Translation Table.

→ frame number = Translation bits.



or page number.

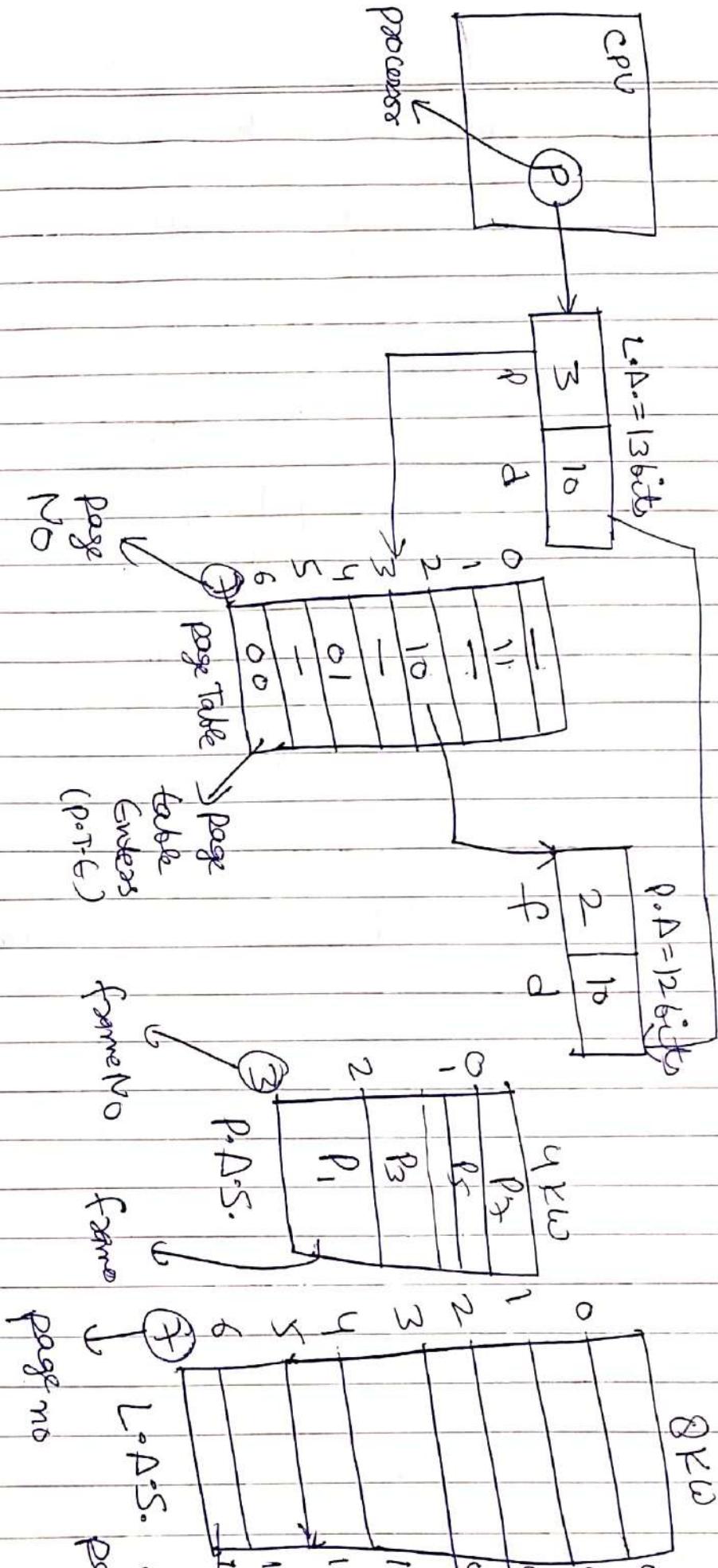
$\rightarrow d = \text{No. of bits required to represent word of a page.}$ or
It is also called as page offset.

Physical address we assumed as a 12 bits.
 $P.A = 12 \text{ bits}$

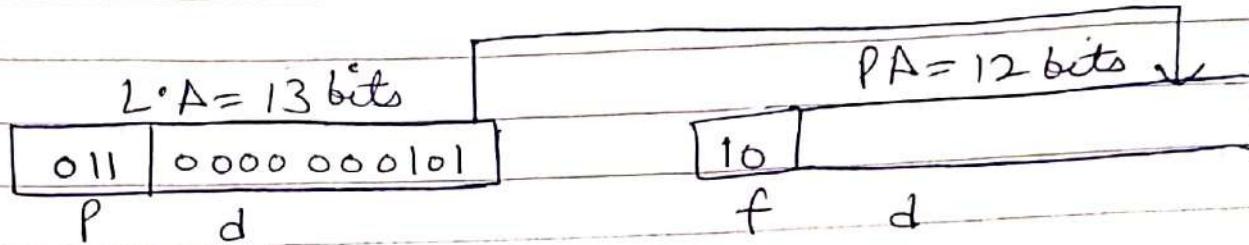
2	10
f	d

$f = \text{No. of bits required to represent frames of PAS}$
OR
frame No.

$d = \text{No. of bits required to represent word No. of a frame or frame offset.}$



Ex:-



Important points :-

- 1.) whenever the process is created, paging will be applied on the process and page table will be created. And The base address of the page table will be stored in P.C.B.. (process Control Block).
- 2.) Paging is with respect to every process and every process will have its own page table.
- 3.) Page tables of the processes will be stored in the main memory.
- 4.) There is no external fragmentation in the paging.
- 5.) The internal fragmentation exists in the last page, and internal fragmentation in the paging is considered as a " $\frac{p}{2}$ ", where p is the page size.

(3) Process size = 16 KB
 Page size = 3 KB
 No of pages = $\frac{16 \text{ KB}}{3 \text{ KB}} = 5.33 = 6$ ✓
 (Internal fragmentation on)

6.) Maintaining "overhead" the page table is considered as for the system.

NOTE:- Windows practically follows page size of 4 KB.

NOTE:- The Page table entry definitely contains frame Number, but sometimes along with the frame No, it may also contain additional bits like

- 1.) Valid / invalid bit or present bit
- 2.) Dirty bit / modified bit.
- 3.) Reference bit.
- 4.) Page protection bits or permission bits.

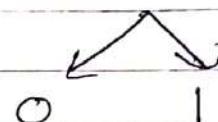
P.T.E = Page table Entry

F.No.	V/I or P	D or m	R	P.P
Frame number	Valid / invalid bit or present bit	Dirty bit or modified bit	Reference bit	Page protection bits or permission bits

frame number (or) is also called translation bits.

Valid/ Invalid or present bit :-

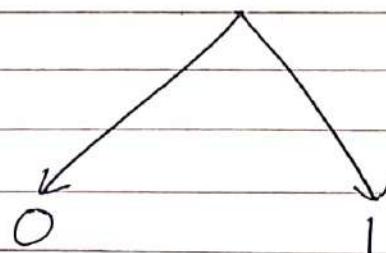
These bits are used to identify whether the page is currently available or not available in the main memory.



Not present in the main memory	present in the M.M.
--------------------------------------	---------------------------

Dirty Bit or modified bit :-

This bit is used to identify whether the page is modified or not modified by the processor while accessing the page as part of the execution.



Not modified	Modified.
--------------	-----------

Reference Bit :-

This bit is used to identify, how many times the page is referred by the processor while execution.

PAGE protection or permission bits :-

These bits are used for the protection and security from unauthorized access.

$$1 \text{ MB} = 10^6 \text{ B} \quad 2^{27} = 2^3 \times$$

Q1.) Consider a system which has logical address of 27 bits. And Physical address of 21 bits and page size is 4 KB. Memory is word Addressable, then calculate No. of pages & No. of frames

$$\begin{aligned}\text{logical A.S.} &= 2^{27} \\ &= 2^{20} \times 2^7 \\ &= 128 \text{ MB}\end{aligned}$$

$$\begin{aligned}\text{Physical A.S.} &= 2^{21} \\ &= 2^{20} \times 2^1 \\ &= 2 \text{ MB}\end{aligned}$$

$$\begin{aligned}\text{No. of pages} &= \frac{128 \text{ MB}}{4 \text{ KB}} \\ &= \frac{128 \times 2^{10}}{4 \text{ KB}} \\ &= \frac{128 \times 2^{10}}{2^2 \times 4} \\ &= \frac{32 \times 2^{10}}{2^{15}} = 32 \text{ K}\end{aligned}$$

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

$$= 2^{11-2}$$

$$= \boxed{2^9} = 512$$

Q.) Consider a system which have No of pages as 8K, page size is 16KB. Memory is byte Addressable. Physical Address is 22 bits, then calculate logical address & No. of frames?

Ans.)

$$\text{Physical A.S} = 2^{22}$$

$$\text{Page size} = 2^4 \times 2^{10}$$

$$\text{Logical Address Space} = \text{page size} \times \text{No of pages}$$

$$= 16\text{KB} \times 8\text{K}$$

$$= 2^{13} \times 2^{14}\text{B} = 2^{27}\text{B}.$$

LA = 27 bits.]

$$\text{No. of frames} = \frac{2^{22} \text{ byte}}{2^{14} \text{ byte}} = 2^8$$

= 256

a.) Consider a system which has

$$\text{LAS} = 128 \text{ MB},$$

$$\text{PA} = 24 \text{ bits}$$

Memory is word Addressable. The PAS is divided into 8K frames. Then what is the page size & how many pages in the LAS?

~~LA~~

$$\text{PAS} = 2^{24} \cancel{\text{bits}} \cdot \text{Words}$$

$$=$$

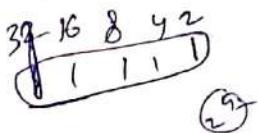
$$\frac{\text{No. of frames}}{2^{24}} = \text{frame size}$$

$$\Rightarrow \frac{2^{24}}{2^{13}} = \text{frame size}$$

$$= \boxed{2^{11}} = \text{frame size.}$$

$$\text{No. of pages} = \frac{2^7 \times 2^{20}}{2^{11}} = 2^{16} \quad \checkmark$$

$$= 64 \text{ K.}$$



Q.) Consider a system which logical address of 32 bits and $PAS = 64MB$.

Memory is byte addressable & Page size is 4 KB. Then what is the approximate size of page table in bytes.

- a.) 1 MB
- b.) 2 MB
- c.) 4 MB
- d.) 8 MB

~~$$\begin{aligned}
 L.A.S. &= 2^5 \\
 P.A.S. &= 2^{12} \\
 P.T.E.S. &= 2^{26} \\
 &= 2^{26} \text{ bytes.}
 \end{aligned}$$~~

$$\begin{aligned}
 \text{No. of frames} &= \frac{PAS}{\text{frame Size}} \\
 &= \frac{64MB}{4KB} \\
 &= \frac{2^{26}}{2^{12}} = 2^{14}
 \end{aligned}$$

14 bits are required

Page table size = No. of entries in the PT * Page table entry size.

$$= \left(\frac{2^{32}}{2^{12}} \right) \times \text{F.NO}$$

$$\begin{aligned}
 &= 2^{20} \times 14 \text{ bits} \\
 &= 2^{20} \times 2 \text{ bytes}
 \end{aligned}$$

$$2^9 = 4 \cdot 2^4$$

Q.) Consider a system which have page table with 4 K entries & logical address is 29 bits then what's the physical address, if the system has 512 frames.

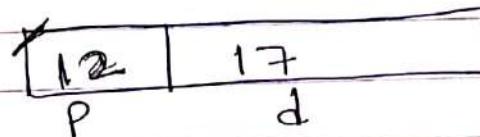
$$\text{4K entries} = 2^2 \times 2^{10}$$

$$\text{No of pages} = 2^{12}$$

$$\text{Page size} = \frac{\text{L.A.S}}{\text{No of pages}}$$

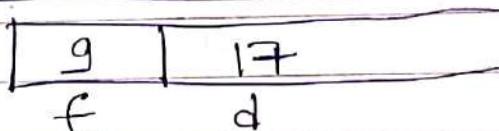
$$= \frac{2^{29}}{2^{12}}$$

$$\text{L.A} = 29 \text{ bits}$$



$$\text{P.A} = 26 \text{ bits}$$

$$512 = 2^9 \\ = 9 \text{ bits}$$



workbook 10

P-107

Q40)

$$1 \cdot A \cdot S = 2^{16} \text{ bytes}$$

$$P \cdot P \cdot S = 2^9 \text{ bytes}$$

$$\text{Page size} = 2^9 \text{ bytes}$$

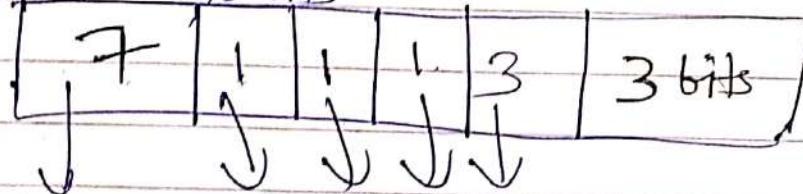
$$\text{Page T.E. size} = 2 \text{ bytes.} = 16 \text{ bits}$$

Page Table = ? $\left(+ \text{ offset + dirty vald} \right)$

$$\frac{\text{No. of frames}}{\text{size}} = \left(\frac{2^{16}}{2^9} \right) \times 2 \text{ bytes}$$

$$\begin{aligned} &= 2^7 \times 2 \\ \frac{\text{No. of frames}}{\text{Page table size}} &= \frac{2^8}{2^7} \text{ bytes} \\ &= 2^8 \times 2^3 \text{ bits} \\ &= 2^{11} \text{ bits} \\ &= 2 \text{ KB.} \end{aligned}$$

$$P.T.E = 16 \text{ bits}$$



F.N.

3 bits

GATE-2016 (2 Marks)

$$(P.T.E.S = \frac{LAS}{PS})^{2015}$$

Q.) Consider a system

$$L.A = 40 \text{ bits.}$$

$$\text{Page size} = 16 \text{ Kb.}$$

Memory is byte addressable. The page table entry requires 48 bits. Then what is the page table size in Megabytes.

Solve:

$$P.T.E. \quad \underline{48 \text{ bits.}}$$

Page Table size = No of frames \times Page table entry size.

$$= 48 \text{ bits} \times \frac{2^{40} \text{ bytes}}{16 \text{ Kb}}$$

$$= \cancel{6 \text{ bytes}} \times \frac{2^{40} \text{ bytes}}{16 \times 2^{10} \times 8}$$

$$= \frac{2^{40} \times 6}{2^{10} \times 2^4 \times \cancel{8}}$$

$$= \frac{2^{40-14} \times 6}{\cancel{2^4}}$$

$$= 6 \times 2^{26}.$$

$$= 6 \times 64 \times MB$$

$$= 384 \text{ MB.}$$

10 bits
23 bits

Q.) Consider a system which has

$$LAS = PAS = 'S' \text{ bytes.}$$

& Page size is ' P ' bytes.
Page T.E. Size = ' e ' bytes.

then what is the optimal value of page size by minimising overhead of maintaining Page table size & internal fragmentation in the paging?

- a.) $P = \sqrt{2se^2}$
- b.) $P = \sqrt{2s^2e}$
- c.) $P = \sqrt{2se}$
- d.) $P = \sqrt{2(se)^2}$

$$\text{page size} = \frac{\text{PAS}}{\text{No of frames}}$$

$$= \frac{S \text{ bytes}}{P}$$

$$\text{No of frames} = \frac{\text{PAS}}{\text{page size}} = \frac{S}{P}$$

$$\frac{P}{S} \times \frac{S}{\text{No of frames}}$$

Memory overhead = P.T. size + I.F. in paging.

$$= \text{No. of entries} \times \text{entry size} +$$

$$= \left(\frac{S}{P} \right) \times e + \frac{l}{2}$$

$$= \frac{d}{dp} \left(\left(\frac{S}{P} \right) \times e + \frac{l}{2} \right) = 0$$

$$\Rightarrow \left(-\frac{Se}{P^2} + \frac{1}{2} \right) = 0$$

$$\Rightarrow P^2 = 2Se$$

$P = \sqrt{2Se}$

PERFORMANCE OF PAGING:-

The Main memory access time = 'M'

The page tables are stored in the main memory.

Then the formula for effective memory access time

$$M \cdot M \cdot A \cdot T = 'M'$$

$$E \cdot M \cdot A \cdot T = 2M$$

1 M for

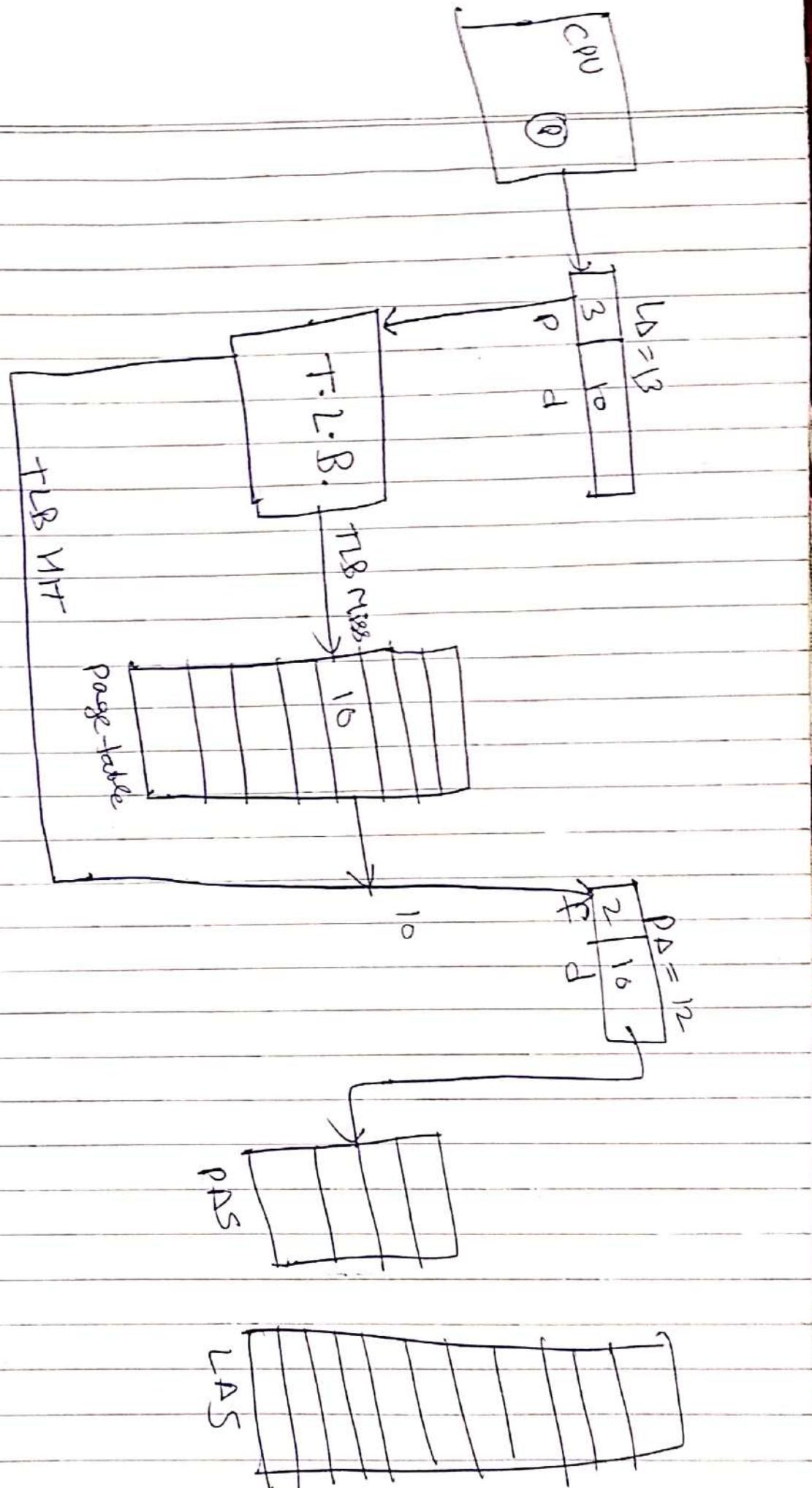
1 M for Actual time.

The T.L.B. (translation look aside buffer) is added to improve the performance of Paging.

→ TLB is a hardware device implemented using associative registers.

→ TLB access time will be very less compare to main memory access time.

→ TLB contains frequently referred page nos & the corresponding frame numbers.



TLB access time = 'c'.
TLB hit ratio = 'x'

then the formula effective memory access time.

$$E.M.A.T = x(c+m) + (1-x)(c+2m)$$

Q.) Consider a system which has m.m.a.t of 100ns. TLB access time is 20ns, TLB hit ratio is 95%. Then what is effective memory a.t with TLB & without TLB?

Ans.)

$$m.m.a.t = 100\text{ns}.$$

$$TLB \text{ access time } c = 20\text{ns}.$$

$$x = 95\% = 0.95$$

with
TLB

$$\begin{aligned} &= 0.95(20+100) + (1-0.95)(20+200) \\ &= 0.95(120) + (0.05)(220) \\ &= \underline{125\text{ ns}}. \end{aligned}$$

without
TLB

$$\begin{aligned} E.M.A.T &= 2M \\ &= 2 \times 100\text{ns} \\ &= 200\text{ns}. \end{aligned}$$

Q.) How much hit ratio is required to reduce effective M.A.T from 300ns without TLB to 250 ns with TLB? The TLB access time is 60 ns.

Ans.)

$$E.M.A.T = 2M$$

$$2M = \cancel{2} \times 300$$

$$M = \cancel{2} \times 150$$

$$E.M.A.T = x(c+m) + (1-x)(c+2m)$$

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

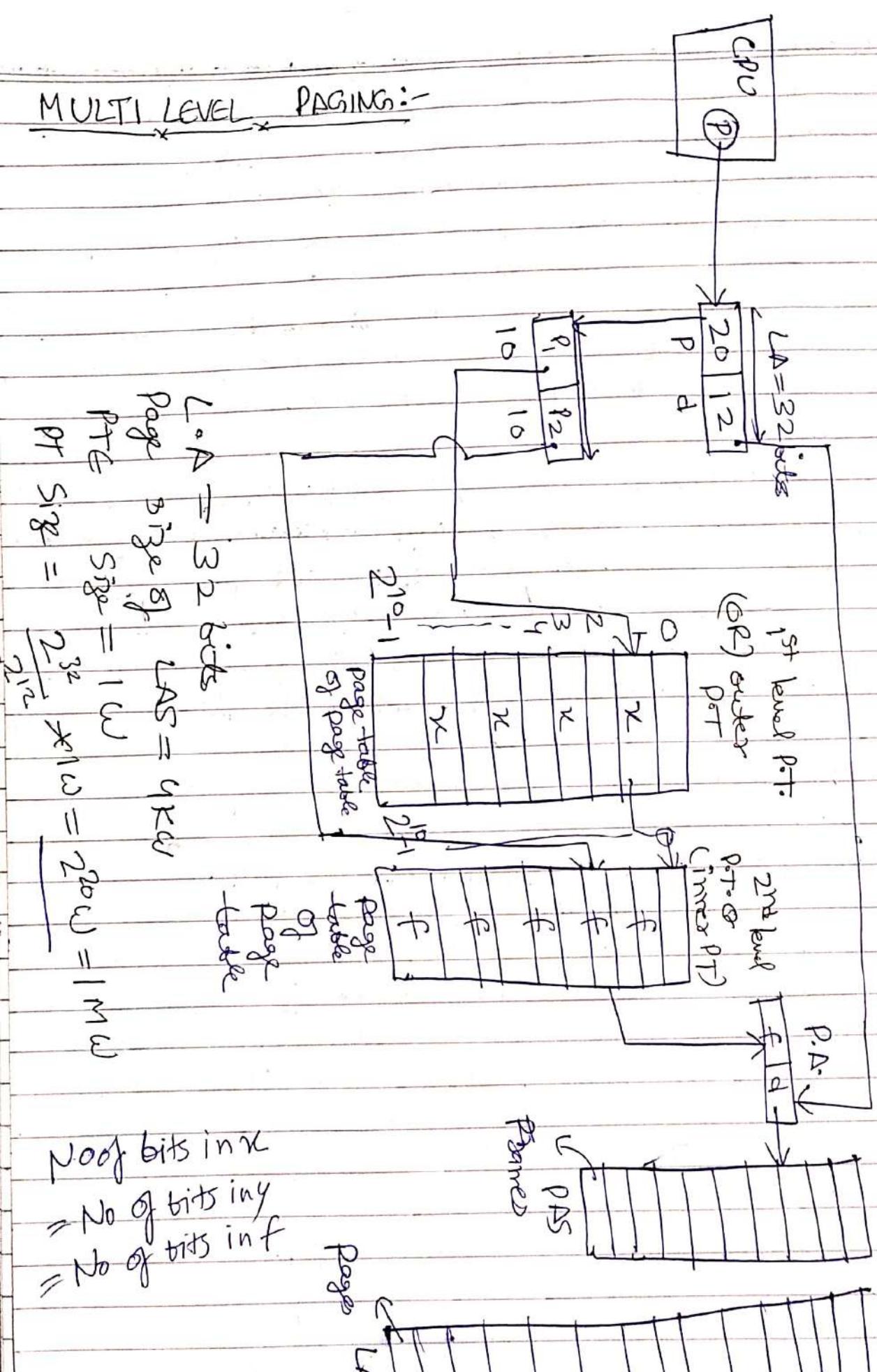
$$250 = x(210) + (1-x)(60 + 300)$$

$$x = 0.733$$

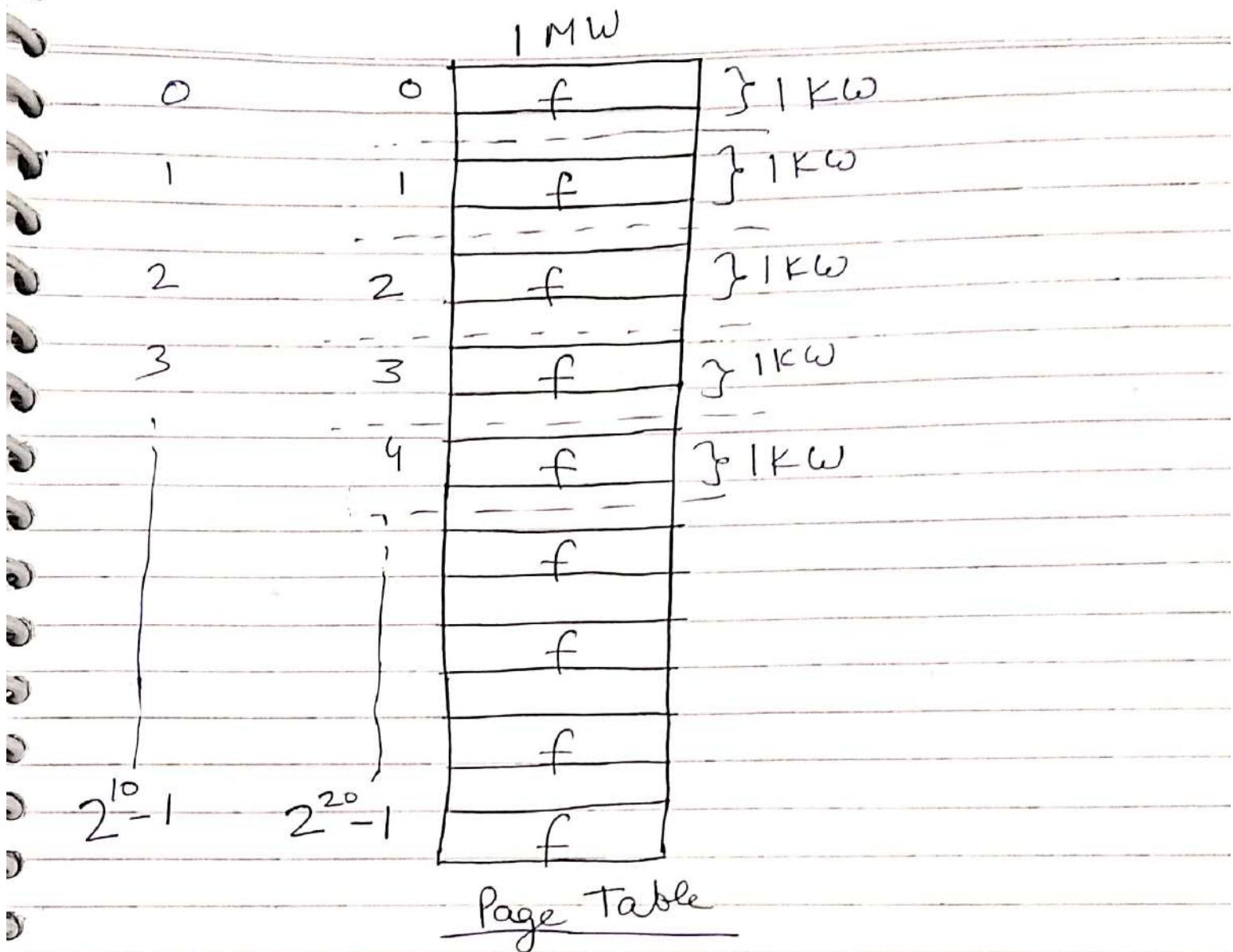
Q.) Consider a system which has logical address of 32 bits and Page size of LAS = 4 KW and memory is word addressable and page table entry size is one word. Then what is the page table size?

Solve.)

MULTI LEVEL PAGING:-



$\kappa = \text{Base Address of Page of P.T.}$
 Page Size of P.T. = 1 KB
 κ = Page Table of Task = 1 MB



- To avoid the overhead of bringing large size page table into memory, the concept of multilevel paging will be implemented.
- In the multilevel paging, paging will be applied on the page table and instead of bringing the entire page table into memory the pages of page table will be brought into memory.

→ P_1 is no of bits required to represent pages of page table or page no. of page.

$$\boxed{\text{No. of pages on P.T.} = \frac{2^{20}W}{2^{10}W} = 2^10 = 1K.}$$

$\rightarrow P_2$ = No. of bits required to represent word no. of page of page table or (page offset of page table).

Ans 10s
D

Q.) Consider a system which has 2 level paging applicable. The Page table has divided into 2^k pages and each page have $4K$ entries. The memory is word addressable. The ~~PAS~~ = 64 MW , which is divided into 8 K frames. Page table entry size in both the levels is 2 words. Then calculate

- i.) Length of logical address
- ii.) Length of physical address
- iii.) Ist level page table size
- iv.) IInd level Page table size
(Page of page table).

$$\text{P.T. size} =$$

$$\text{No. of pages} = 2^k$$

$$\text{I}^{\text{st}} \text{ level pages} = 4K \text{ entries.}$$

no. of pages

$$\text{Page table entry size} = 2^w$$

$$\boxed{\text{Frame size}} \neq \cancel{\text{PAS}} = \frac{2^6 \times 2^{20} w}{2^3 \times 2^{10}} = 2^3 \times 2^{10} = 8Kw. \\ = 2^{13} w.$$

$$(i.) \cancel{\text{LAS}} = 2^R \times 2^{13}$$

$$= 2^{14} \cancel{62}$$

$$\boxed{\text{LA}} = \boxed{14}$$

$$(ii.) \cancel{\text{PA}} = 64\text{ MLO}$$

$$\rightarrow 2^{26} w$$

$$\boxed{\text{PA}} = \boxed{26}$$

$\frac{2^w}{2^w}$

①

(iii.)

$$= 4K \times 2^{12} - 2K \times$$

$$= \cancel{2K \times 2^{12}}$$

$$= \cancel{2^1}$$

(iv.)

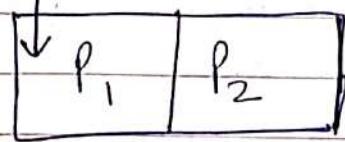
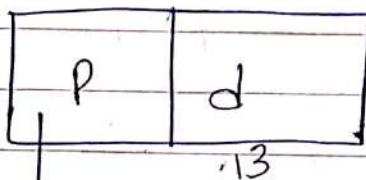
$$= 4K \times 2^{12}$$

$$= 2^{13}$$

①

LA

$$= 11 + 12 + 13 = \underline{36}$$



②

$$PAS = 64M\omega$$

$$= 2^{26}\omega$$

$$PA = 2^{26}\omega.$$

iii.)

$$= \cancel{2^1} \times \cancel{2^1}$$

$$1^{st} \text{ level P.T.} = 2^{p_1} \times \text{PTE Size}$$

Size

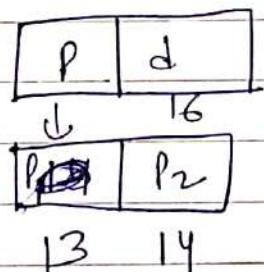
$$= 2^{11} \times 2\omega$$

$$= 2^{12}\omega$$

4.) 2nd level P.T. size = $2^{P_2} \times P.T.E.\text{-size}$
 (page of P.T)
 $= 2^{12} \times 2^4$
 $= 2^{16} = 8\text{K}\omega.$

Q.) Consider a system which has 2 level paging applicable. The page table has divided into 8 K pages. And each page having 16 K entries. Memory is byte addressable. The PAS is 256 MB which is divided into 64 KB frames. The page table entry size in both the levels is 32 bits then calculate.

- 1.) Length of logical address
- 2.) Length of physical address
- 3.) Ist level page table size
- 4.) IInd level page table size
 (page of page table).



frame size = 64KB
 $= 2^{16}$

$$\begin{aligned} LA &= 13 + 14 + 16 \\ &= 43 \end{aligned}$$

2.) PAS = 256 MB

$$\text{PAS} = 2^{28}$$

3.) 1st level Page Size = $2^{P_1} \times \text{PTEsize}$

$$= 2^{13} \times 32 \text{ bits}$$
$$= 2^{13} \times 2^2$$
$$= 2^{15} \text{ w}$$
$$= \cancel{\text{32KB}} \quad 32\text{KB}$$

4.) IInd level Page Size = $2^{P_2} \times \text{PTEsize}$

$$= 2^{14} \times 2^2$$
$$= 2^{16} \text{ w}$$
$$= \cancel{64\text{KB}} \quad 64\text{KB}$$

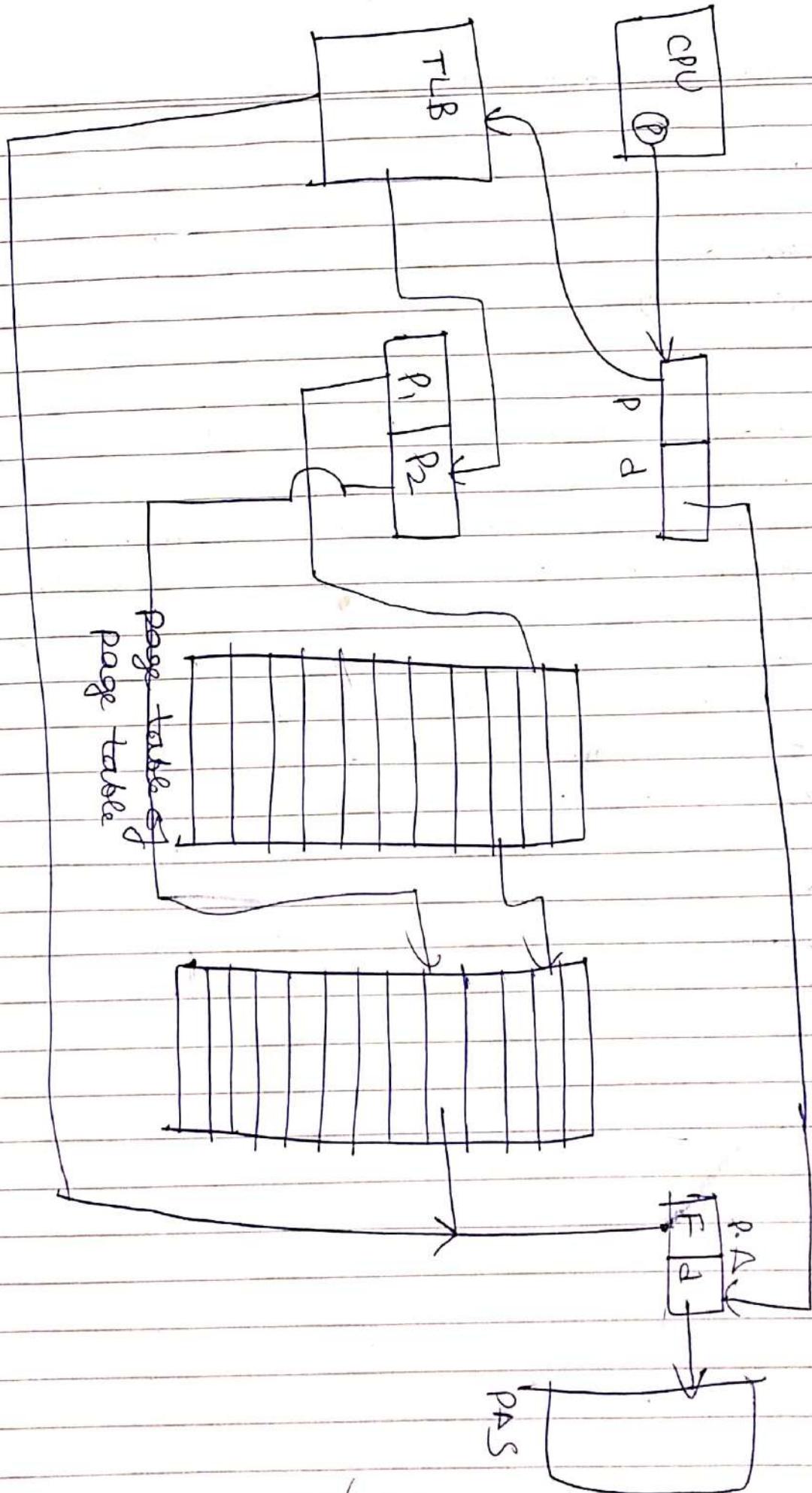
PERFORMANCE OF TWO LEVEL PAGING :-

$$\text{MMAT} = 'M'$$

The page tables are stored in the main memory, then the formula for

$$\boxed{\text{E.M.A.T} = 3M}$$

- The TLB is added to improve the performance of Paging.
- The TLB contains frequently referred page numbers and corresponding frame numbers.



→ TLB ACCESS TIME = 'c'

→ TLB HIT RATIO = 'x'.

Then the formula for effective M.A.T

$$EMAT = x(c+m) + (1-x)(c+3m)$$

n level paging

$$EMAT = x(c+m) + (1-x)(c + (n-1)m)$$

IMPORTANT POINTS:-

- In the multilevel paging, when the paging applied on the page tables, then the last page table, which we get is called as first level page table (I^{st} Level).
- In a multilevel paging, when the paging applied on the page tables, then I^{st} level page table entry contains base address of II^{nd} level page table, II^{nd} level page table entry contains base address of III^{rd} level page table and so on. The final page table entry contains frame no. of a actual page.

No of bits in X = No of bits in Y = No of bits in F

- In the multilevel paging, when the paging applied on the page table, then whatever may be the levels of paging, all the page tables "Page of Page table" will be stored in a main memory.
- In the multilevel paging, when the paging applied on the page tables, then whatever may be the levels of paging. All the page table entries contains frame no.
- If the page size is not mentioned in the problem then generally page size will be same in all the places (levels).

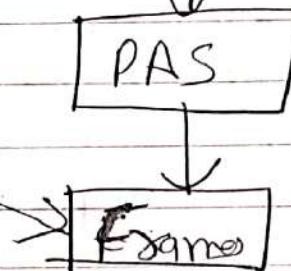
The page tables are stored in the

Main memory

No of bits in X

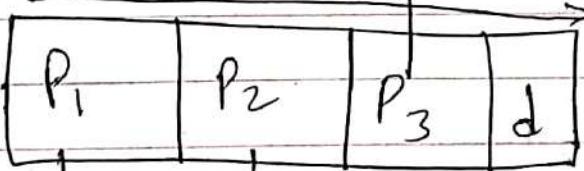
=
No. of bits in Y

=
No of bits in F

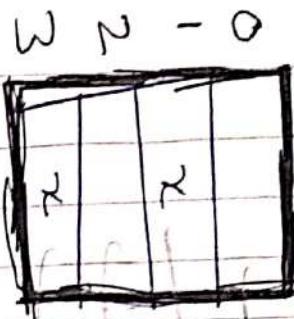


L.A.

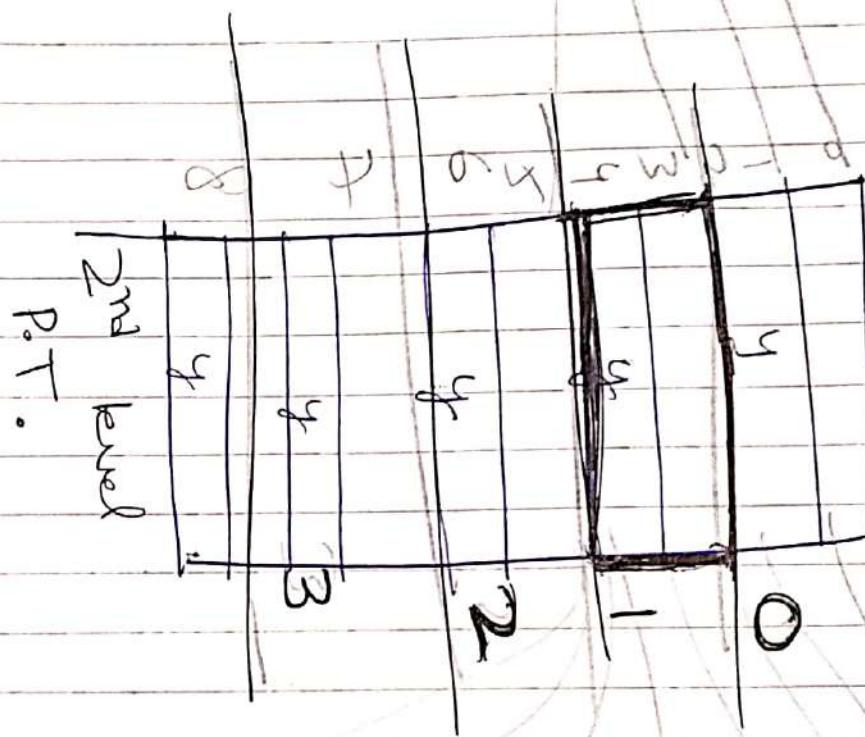
→ 3rd level Page indexing



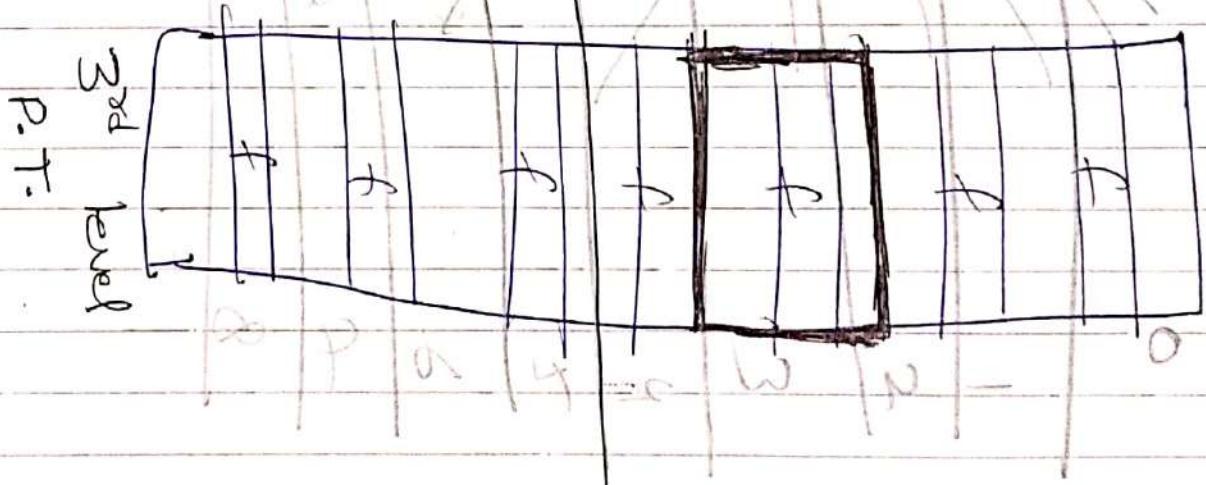
→ 2nd level P.T. index



1st level
P.T.



2nd level
P.T.



3rd level
P.T.

P-106
workbook

(2)

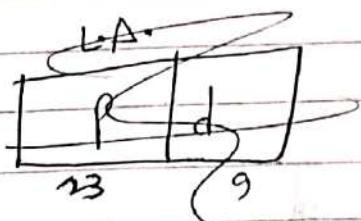
$$\frac{32}{2} =$$

$$\frac{1024}{2^9}$$

Q25.) ② logical address = 32 bit
= 4 bytes.

~~1024~~ pages ~~size~~ = $1024 \text{ bytes. } 2^9$

Page table entry = 4 bytes.



$$1024 = 4 \text{ bytes} \times \text{No. of entries}$$

$$\frac{1024}{4} = \text{No. of entries}$$

$$256 = \text{No. of entries.}$$

③

P.T size \leq Page size

1st time paging :-

$$\begin{aligned}\text{P.T size} &= \left(\frac{2^{32}}{2^{10}} \right) \times 4B \\ &= 2^{22} \times 2^2 B \\ &= 2^{24} B \\ &= 16 \text{ MB}\end{aligned}$$

IInd time paging :-

$$\begin{aligned}
 \text{P.T. size} &= \left(\frac{16 \text{ MB}}{1 \text{ KB}} \right) \times 4 \text{ B} \\
 &= \frac{2^{24}}{2^{10}} \times 2^2 \text{ B} \\
 &= 2^{16} \text{ B} \\
 &= 64 \text{ KB}.
 \end{aligned}$$

IIIrd time paging :-

$$\begin{aligned}
 \text{P.T. size} &= \frac{64 \text{ KB}}{1 \text{ KB}} \times 4 \text{ B} \\
 &= 64 \times 4 \text{ B} \\
 &= 2^8 \text{ B} \\
 &= 256 \text{ bytes. } (\underline{\text{9th level P.T. size}})
 \end{aligned}$$

16MB \leq 1KB	X
64KB \leq 1KB	X
256B \leq 1KB	✓

(C) is right answer.

2marks

Grade 2013



P.A = 32²

Q1.) Consider a system which has logical addresses of 46 bits, P.A = 32 bits.

Memory is byte addressable.

The Page table entry size is 32 bits.

The OS uses 3 level paging for logical to physical address translation and 1st level page table size is exactly same as page size. What is the page size?

a.) 2 Kb.

b.) 4 Kb.

c.) 8 Kb. ✓

d.) 16 Kb.

Solve:-)

$$l.a = 46 \text{ bits}$$

Page table entry size = 32 bits.

No of entries in the PT =

1st time paging :- Assuming page size κ

$$\text{P.T. size} = \frac{2^{46}}{\kappa \text{ bytes}} \times 32 \text{ bits}$$

$$= \frac{2^{46} \times 4 \text{ bytes}}{\kappa \text{ bytes}} = \frac{2^{48}}{\kappa} \text{ pages.}$$

2nd time paging :-

$$= \frac{2^{48}/\kappa}{\kappa \text{ bytes}} \times 4$$

$$= \frac{2^{48} \times 2^2}{\kappa^2} = \frac{2^{50}}{\kappa^2}$$

3rd time Paging :-

$$\Rightarrow \frac{2^{50}}{\pi^2} \times 4 \text{ bytes}$$

$$\boxed{P.T.\text{-size} = \frac{2^{52}}{\pi^3}} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{9st level Page-Table size.}$$

1st level page-table size = Page size

$$\frac{2^{52} B}{\pi^3} = x B$$

$$\begin{aligned} \pi^4 &= 2^{52} \\ \pi^4 &= (2^{13})^4 \end{aligned}$$

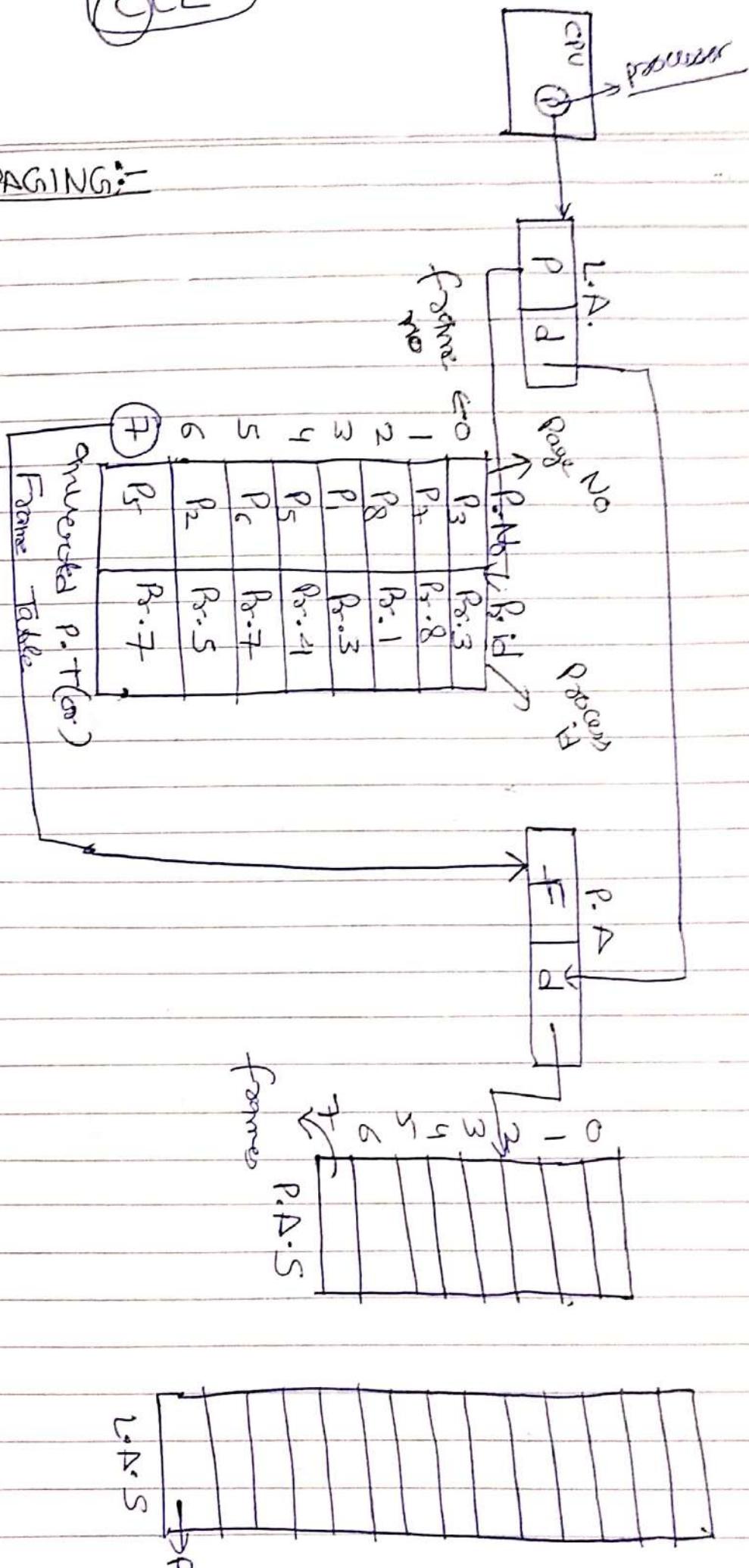
$$\pi = 2^{13} B$$

$$\boxed{\pi = 8 KB.}$$

option (c.)

(GCL)

INVERTED PAGING :-



- To avoid, the overhead of maintaining page table for every process, the concept of inverted paging will be implemented.
- In the inverted paging, one page table will be maintained for all the processes.
- No. of entries in the inverted page table is same as the no. of frames in P.A.S.
- The memory required to maintain page table for the process will be less, but the searching time for a corresponding page of a process will be more.

Eg:-

(P.No, Pr.id)
(P ₅ , Pr ₇)

Q.) Consider a system which has logical address

of 34 bits,

$$P.A = 2^9 \text{ bits}$$

$$\text{Page size} = 16 \text{ KB}$$

$$\text{Page table entry size} = 8 \text{ B.}$$

Memory is byte addressable.

Calculate

- i.) Conventional Page table size.
- ii.) Inverted Page table size.

(Solve)

$$= 2 \times \frac{2^{34}}{2^9}$$

$$= 2 \times \cancel{\frac{2^{34}}{2^9}} \quad 2 \times 2^9$$

$$= \cancel{2^{25}} \quad 8 \text{ MB.}$$

ii.)

$$\Rightarrow 2 \times \text{No. of frames}$$

$$\Rightarrow 8 \text{ B} \times \left(\frac{2^{23}}{2^9} \right)$$

$$\Rightarrow 2^{15} \times 8$$

$$= 2^{18} \text{ B}$$

$$= 256 \text{ KB.}$$

P-106

workbook

PA + 12 bits

241

Q23.)

$$L.A = 32 \text{ bits} =$$

$$(P.A)_{RAM} = \frac{\text{page size}}{128 \text{ KB}} = 128 \text{ KB}$$

$$\text{Page table E.S} = 4B$$

Solve:

$$\phi = 4 \times \frac{2^{32}}{2^{12}}$$

$$(q) = \boxed{2^{15} : 1}$$

P-104
workbook

Q5.)

$$\text{logical A} = 4KB$$

L.A - 32 bit

$$PA = 30 \text{ bit}$$

$$\text{Page size} = 4KB.$$

$$P.A + E.S = 20 + 12$$

$$\text{Page No} = x.$$

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

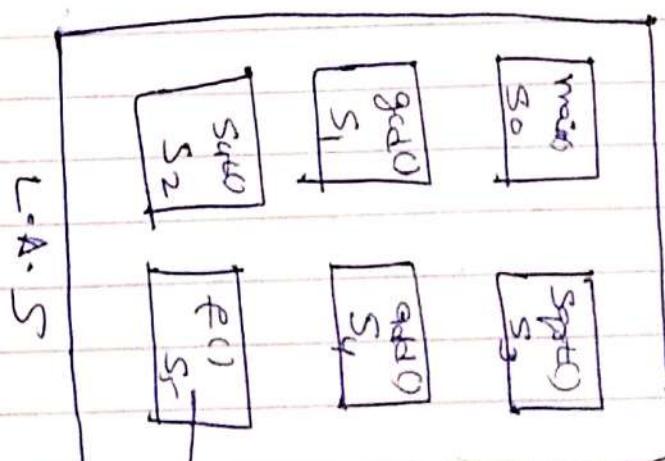
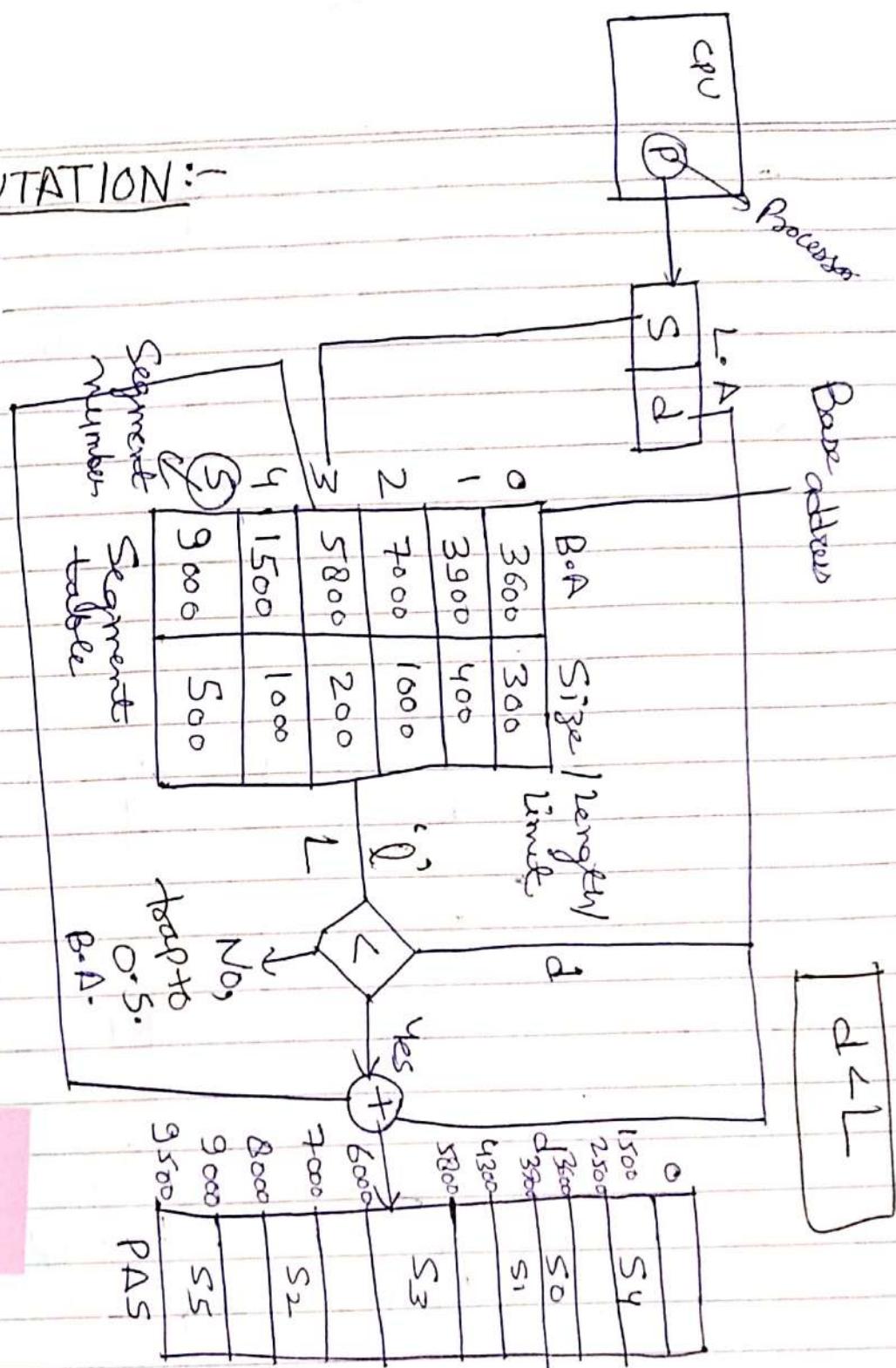
$$\text{No. of frames} = \frac{2^{30}}{2^{12}} = 2^8$$

$$= \text{No. of entries} \times \text{No. of frames}$$

$$= \left(\frac{2^{30}}{2^{12}} \right) \times (P.No + 12 \text{ bits})$$

$$= [2^{30}] \times (20 \text{ bits} + 12 \text{ bits})$$

SEGMENTATION:-



- The Paging does not follow users views of memory allocation.
- To achieve, users view of memory allocation, the concept of segmentation will be implemented.
- In a segmentation, L.A.S will be divided into various segments.
- The segments of LAS will vary in the size.
- Segments of LAS will be brought into P.A.S.
- $S = \text{No of bits required to represent segments of L.A.S or segment number}$
- $d = \text{No of bits required to represent word number of the segment offset}$
- No. of entries in the segment table is same as No of segments in LAS.

Eg :- ① $d=210, L=200$

$$d < L$$

$$210 < 200 \quad \times$$

⇒ Trap to OS

Eg(2) : $d = 190$, $L = 200$

$$\boxed{d < L} - \boxed{190 < 200} \checkmark$$

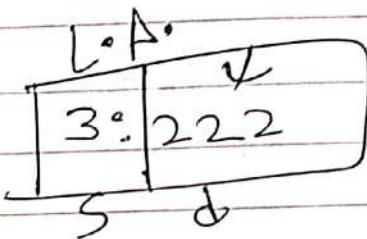
$$B.A + d \Rightarrow P.A.S.$$

$$5800 + 190 \Rightarrow 5990.$$

→ The variable size segments are brought from LAS to PAS, so it is similarly behaving like variable partition scheme. Hence, segmentation still suffers from External fragmentation.

~~Workbook~~

~~Page: 106
Q24.)~~

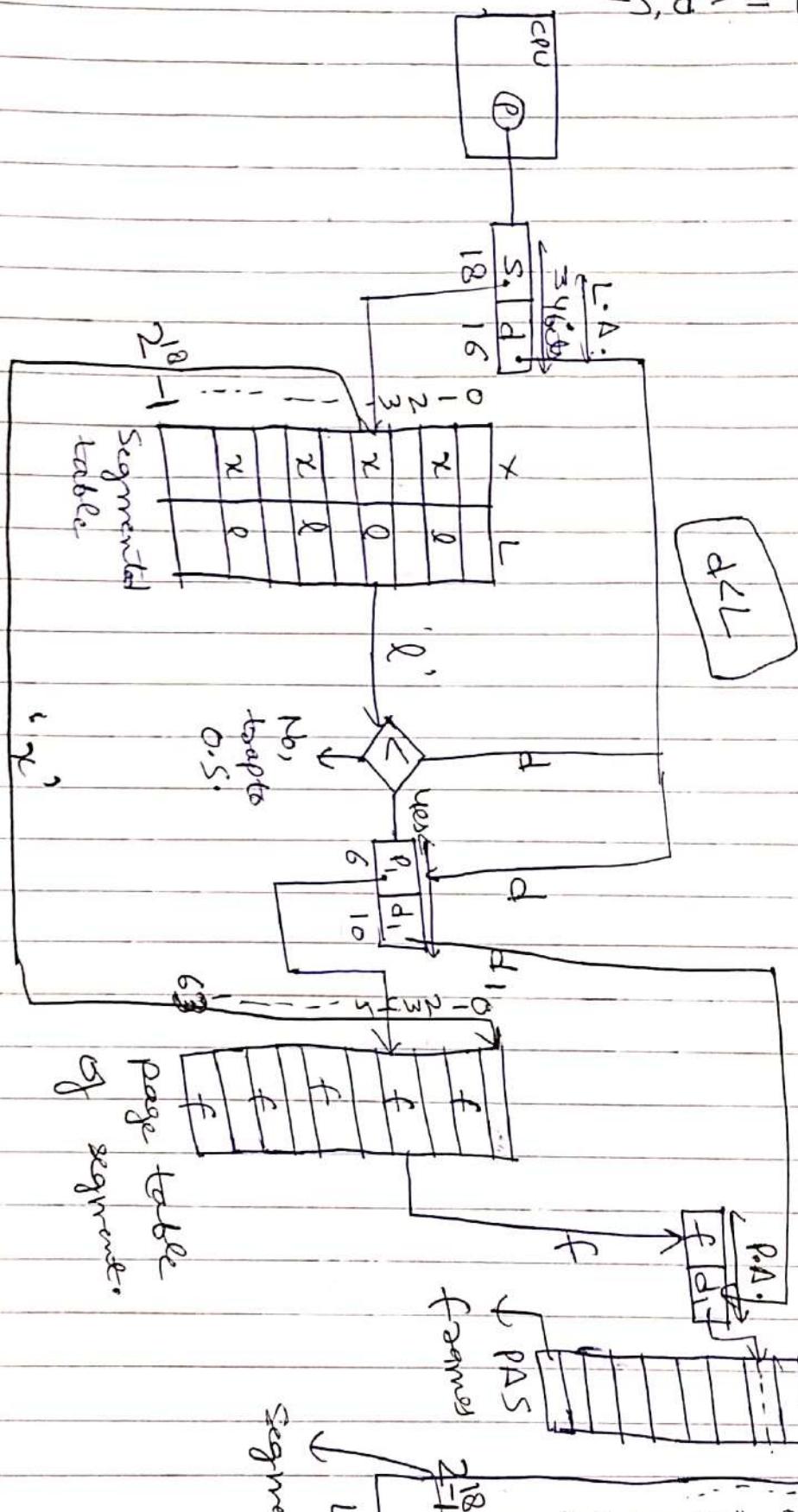


498
222
220 ⑦ 720

SEGMENTED PAGING :-

LA = 34 bits

$\langle S, d \rangle = (18, 16)$



{ Paging on segment, when segment size increases }

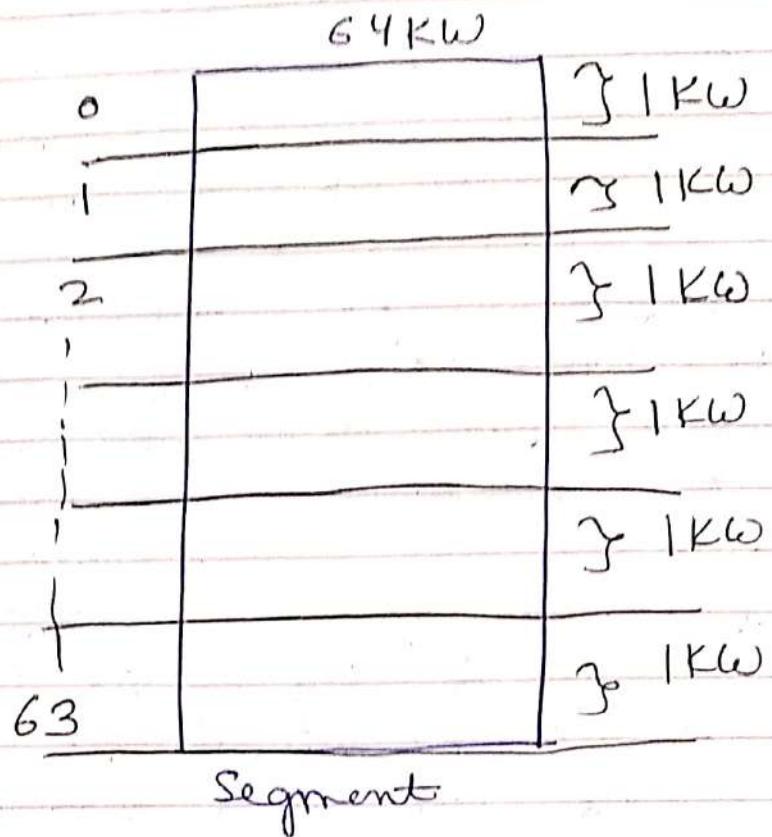
x = Base address of P.T. of respective segment.

Page size of segment = 1 KB.

$$L \cdot A = 34 \text{ bits}$$

$$\langle s, d \rangle = (18, 16)$$

Segment Size = $2^{16} \text{W} = 64 \text{KW}$.



No of Pages of segment = $\frac{\text{Segment Size}}{\text{Size of a page in a segment}}$.

→ To avoid the overhead of bringing large size segment into main memory, the concept of segmented paging will be implemented.

→ In the segmented paging, paging will be applied on the segment and instead of bringing the entire segment of memory, the pages of segment will be

No of entries in the page table of a segment is same as No. of pages on segment.

$P_1 \Rightarrow$ No of bits required to represent pages of segment or page no. of a segment.

$d_1 \Rightarrow$ No of bits required to represent word number of page of segment or page offset of segment.

Page size of segment is same as frame size of PAS.

CCL

Q.) Consider the system using Segmented paging architecture. The segment is divided into 1 K pages. Each page having 512 entries. The memory is word addressable. The segment number requires 17 bits to represent all the segments of LAS and frame number requires 13 bit to represent all the frames of PAS. The page table entry size is 2 words.

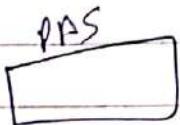
Then calculate

- (i.) Length of logical Address
- (ii.) " Physical Address
- (iii.) Page table size of a segment.

(Ans.) (i.)

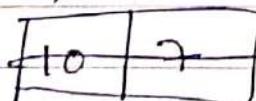
$$1K \text{ pages} = 2^{10}$$

~~S = 10 bits~~



$$\begin{aligned} PA &= 13 \text{ bit} \\ PPS &= 2^3 \end{aligned}$$

$$L \cdot A \cdot S = 13 \text{ bits}$$



$$\begin{aligned} S &= 10 \text{ bits} \\ D &= 7 \text{ bits} \end{aligned}$$

$$= 2^9$$

$$\begin{aligned} \text{No. of Pages in a } \cancel{\text{page}} &= 512 \\ \text{segment} & \end{aligned}$$

$$\boxed{\cancel{LAS} = 2^{17} = 128 \text{ KB}}$$

$$\cancel{LA = 17}$$

$$\frac{P_h}{frame size} = \frac{1000}{16}$$

LAS

PA

(ii.)

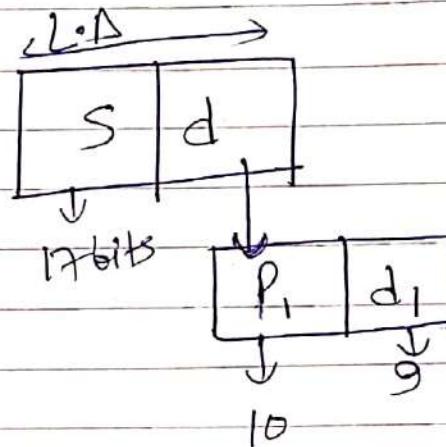
$$PA = 13$$

13	-
----	---

(iii.) Page table size = No. of entries in LAS × Page
size

$$\begin{aligned} &= \cancel{2} \times 2^1 \\ &= \cancel{2} \end{aligned}$$

①



$$17 + 10 + 9 = 36 \text{ bits}$$

②

P·A.

$$\boxed{f \quad d_1} = 22 \text{ bits.}$$

13 9

③ Page table size of a segment

$$\begin{aligned} &= \text{No. of entries} \times \text{entry size} \\ &= \text{No. of pages on segment} \times \text{entry size} \end{aligned}$$

$$= 1K \times 2W$$

$$= 2KW$$

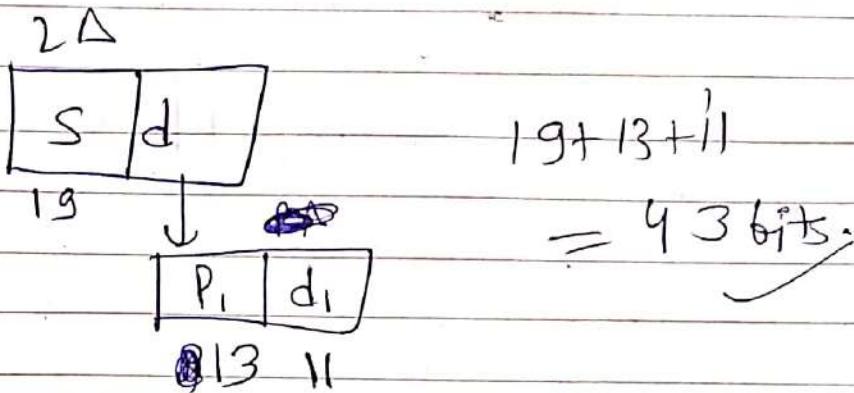
$$2^{10} \quad 2^{3 \times 2^{10}}$$

q.) Consider a system using segmented paging architecture. The segment is divided into 8K pages and each page have 2k entries. The memory is byte addressable. The segment No requires 19 bits to address all the segments of LAS. The PAS is 512KB. And Page table entry size is 8 bits.

Calculate

- (i) Length of 2.A.
- (ii) Length of P.A.
- (iii) Page table size of segment.
- (iv) No. of frames in PAS.

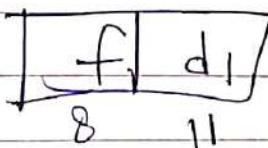
Solve: (i)



$$d_1 = 2^{16} \times 2 \\ = 2^{11}$$

$$\text{PAS} = 2^9 \times 2^{10} \\ = 2^{19}$$

(ii.) P.A



$$\text{PPS} = 19 \text{ bit.}$$

iii) Page table size = No of entries \times P.T.E.S.

$$= \cancel{2^{12}} \cdot \cancel{2^8} \cdot \cancel{2^8} \cdot 2^{13} \times 1$$

$$= \cancel{2^{12}} \cdot \cancel{2^8} \cdot \cancel{2^8} \cdot = 2^{13}$$

$$= 8KB. \checkmark$$

iv)

No of frames in PAS = $\frac{\text{PAS}}{\text{frame size}}$

$$= 2^8. = (\text{using PAS}).$$

$$= 256.$$

workbook
Page 106)

52 - 119 B
8

Q 28.)

$$L.A.S = 2^{16} B$$

$$P.A.S = 2^{16} B$$

pages on
No of segments = equal size segments

= [3 bit]

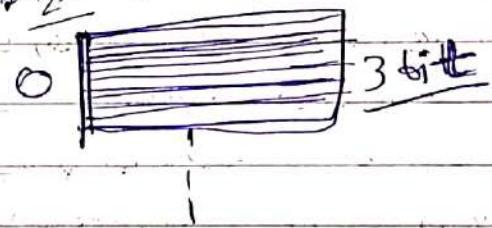
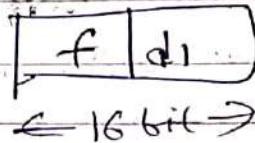
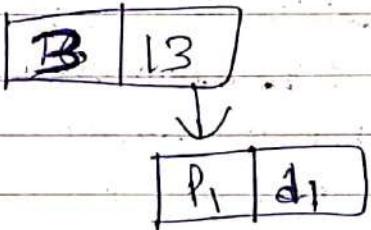


$$P.T.E.S = 2 B,$$

2^{-}

$$L.A. = 16 \text{ Bit}$$

Po A.



$$\text{Segment size} = 2^{13} \text{ Bytes.}$$

8 segments

$$\text{Page size of segment} = 2^{\chi} \text{ Bytes.}$$

$$\text{Page size of segment} = \text{Page table size}$$

$$\text{Page table size of a segment} = \text{No of entries in PT} \times \text{P.T.E size}$$

of segment

= No of Pages on segment \times PT-E. Size

$$= \frac{2^{13} B}{2^x B} \times 2B$$

$$= 2^{14-x} B.$$

Page T. size of a segment = frame size of PAS

$$2^{14-x} B = 2^x B$$

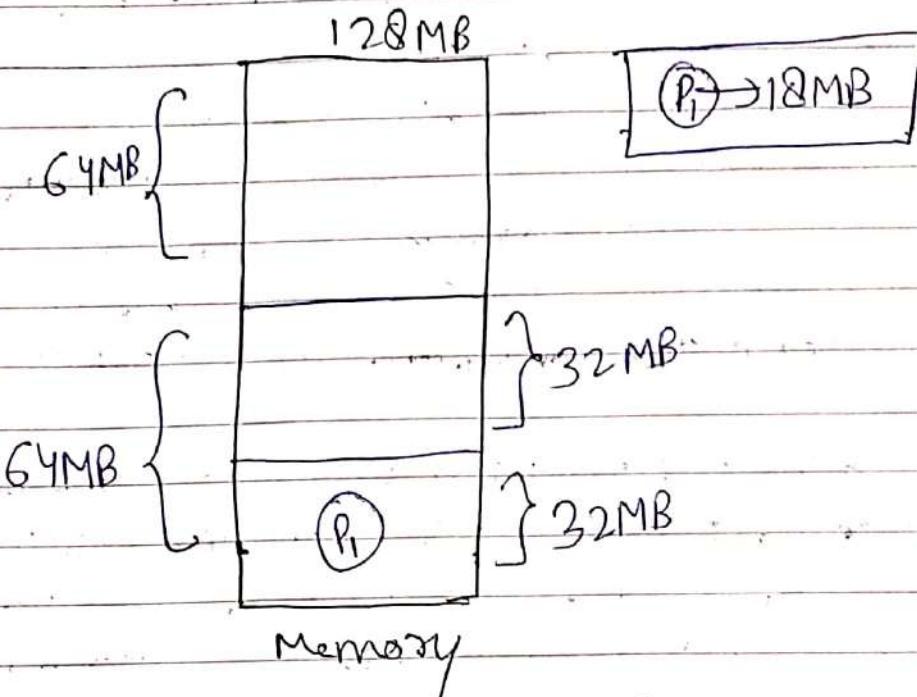
$$2^{14} = 2^{2x}$$

$$2x = 14$$

$$\boxed{x = 7}$$

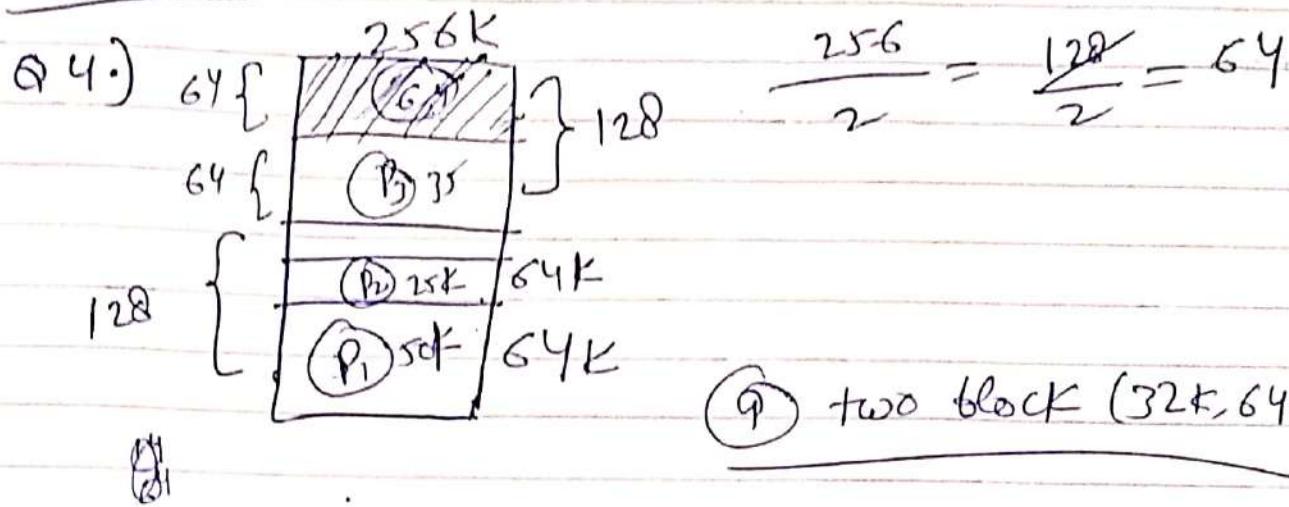
Page size of segment $= 2^x B = 2^7 B$
 $= 128 B.$

Buddy System :-



- In the buddy system, initially, the memory will be single continuous free block.
- Whenever the request by the process arrives, memory will be divided into "two half blocks".
- If the request by the process is too small, then the lower block of the memory is further divided into "two half blocks" again.
- In the buddy system, memory will be allocated from lower blocks to higher blocks.

Workbook
P-105



⑨ two block (32K, 64K)

3.) ⑨ 120K .

Q5.) (b.)