Date

16.06.12

Cosmos

Text Books :- ① Galvin
② Modern OS by Tannenbaum
③ OS by William Stallings

Schedule :-

1. Introduction & background
2. Process Management
   - → process concept
   - → CPU Scheduling
   - → Synchronisation
   - → Concurrent programming
   - → Deadlocks
   - → Threads

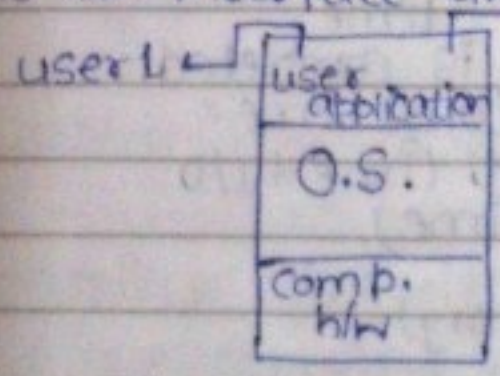
   → 40%
3. Memory Management
   - → RAM chip Implementation
   - → Loading & Linking
   - → Address Binding
     - ↳ paging
     - ↳ multi-level paging
     - ↳ inverted paging
     - ↳ Segmentation
     - ↳ Segmented paging
   - → Virtual Memory

   → 40%
4. File & Device Management
5. Protection & Security

Introduction & Background

→ What is an Operating System :-

It is an interface b/w user & computer hardware.




```
main()
{ int x;
  printf("Hello World
}
```

internally calls write system call in order to communicate with monitor.

System Call :-

It is a request made by the user program in order to

Cosmos

get any kind of service from OS.

- For diff. kind of services, there are diff. kinds of System calls.

→ Operating System can also be called as resource allocator. OS is responsible for resource allocation.

Resources
- H/W type e.g. pointer, memory
- S/W type e.g. files

→ OS acts as government which manages everything inside a computer.

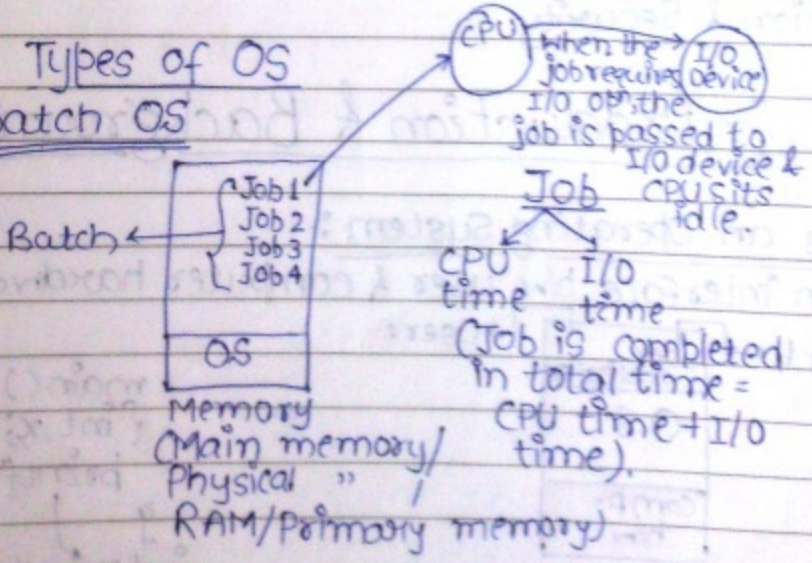
Goals of OS:-

Primary Goal:- convenience (ease of use)
Secondary Goal:- efficiency

- Windows primary goal is convinence & Unix " " " efficiency.

Types of OS

1. Batch OS



★ True In order to execute the job, the job must reside in the main memory.

In Batch OS, if a job requires I/O opn. in b/w, then the CPU will sit idle till that job performs its I/O opn. & comes back to the CPU, i.e. in b/w the CPU won't be allocated to other job.

* The primary responsibility is to place the job into CPU.

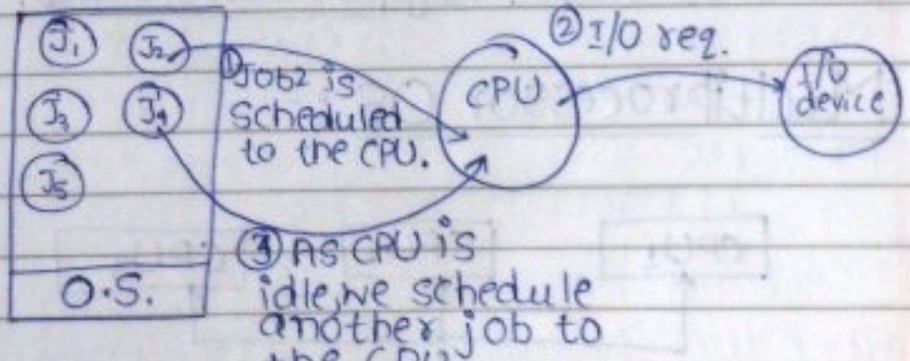
* The other job is scheduled (given to CPU) only when the job is completed completely.

Disadvantages:- CPU utilization is less, so throughput of the system will decrease.

Throughput:- The no. of jobs executed / unit time is called throughput of the system.

e.g. IBM OS/2

2. Multiprogramming OS

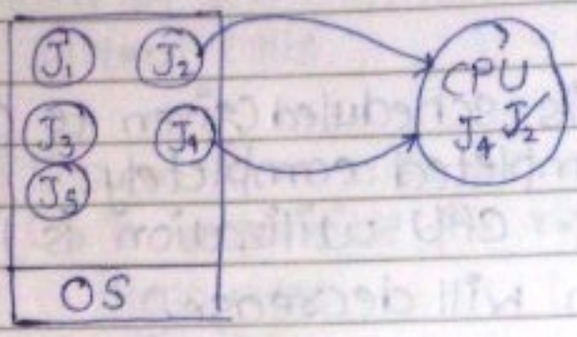


Advantage:- increased CPU utilization & hence increased throughput of the system.

3. Multitasking

The multitasking OS is an extention to Multiprogramming OS. The jobs will be executed on the CPU in time sharing mode.

Cosmos



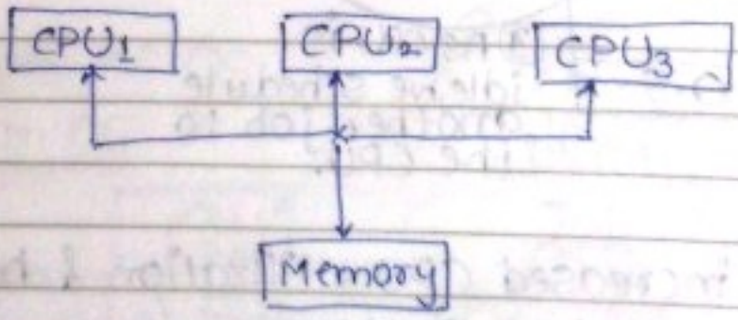
This takes place like:-

→ For a particular amount of time J_1 is executed.

→ When the time expires, Job J_2 preempts J_1 & J_2 is scheduled to CPU.

★ Only one job is executed on the CPU at any point of time. (e.g. Windows, Unix).

Multiprocessor OS :-



★ One system with multiple processors.

• OS is required to support h/w.

Advantages:-

1. Increased Throughput of the System.
2. Reliability: If one CPU fails, the other CPU's will take care that entire system doesn't crashes.

It is also called Fault Tolerance System.

3. It is economical.
e.g. buying 1 system with n CPUs < buying individual systems with 1 CPU each.

Real Time OS :-

The systems which are strict deadly time bound
↓
if an inst^n takes 'n' ns to execute, it should execute in not more than 'n' ns.

✓ Hard Real Time
e.g. missile systems, satellite system
[there should not be any delay at all.]
e.g. no delay is accepted in missile launch.

✓ Soft Real Time
e.g. Banking System
[minor delays are accepted.] e.g. some delay in updation of balance.

★ e.g. RTOS

Process Management

Process Concept :-

Defn. :- program under execution (process).

① The program should reside in the main memory
② It has occupied the CPU to execute the inst^n.

Process has attributes, states & is able to perform various operations.

Attributes :- (context of the process / it is stored in process control block (PCB))
- Process id
- Process state
- Program Counter
- Priority
- General purpose reg.
- List of open files
- List of " devices
- Protection info.

States :-
- New
- Ready
- Running
- Wait (or) Block
- Termination (or)
- Completion
- Suspend Ready state
- Suspend wait (or) suspend block

Operations :-
- Creation
- Scheduling
- executing
- killing/ termination

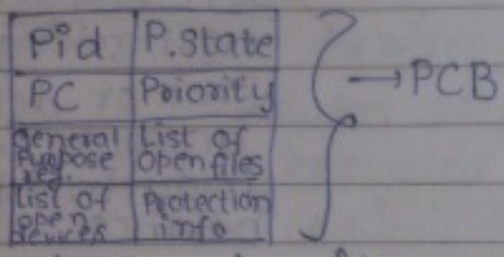
Cosmos

Process id:- It is a unique identification no. which is assigned by OS during process creation.

Process State:- It contains the current state info. where the process is residing.

Program Counter:- The PC contains the address of next instⁿ to be executed.

Priority:- Priority is a parameter which is assigned by the OS at the time of classes creation.



★ every process has its own PCB.

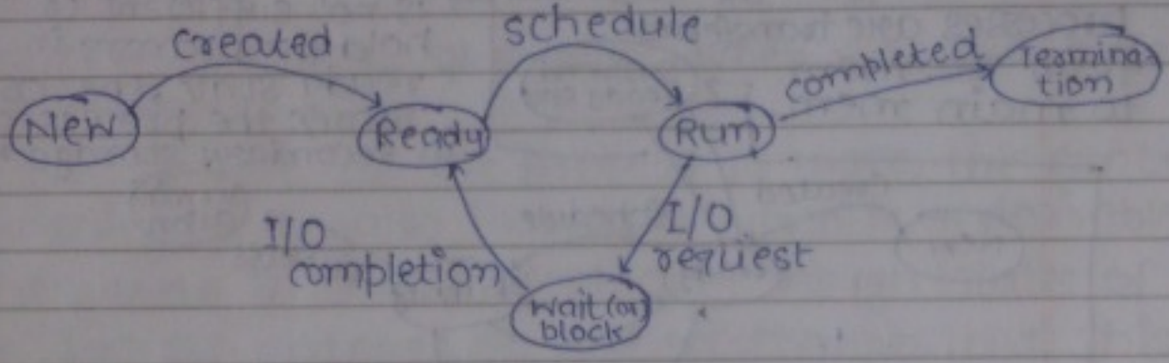
Process State Diagram:-

- **New:-** the process is under creation, & when the process is created it moves to Ready state. [under creation means it is being shifted to the main memory.]
- **Ready:-** in ready states, there are a no. of processes, then we select one process & we schedule it to the running state.
- **Running:-** in this state, CPU executes the instⁿ of the running process. In this state, there is only one process. If the process requires any I/O operⁿ, the process will move to wait or block state.
- **Wait/block state:-** process performs I/O oprⁿ. When I/O operⁿ completes, the process

Cosmos

goes to ready state.

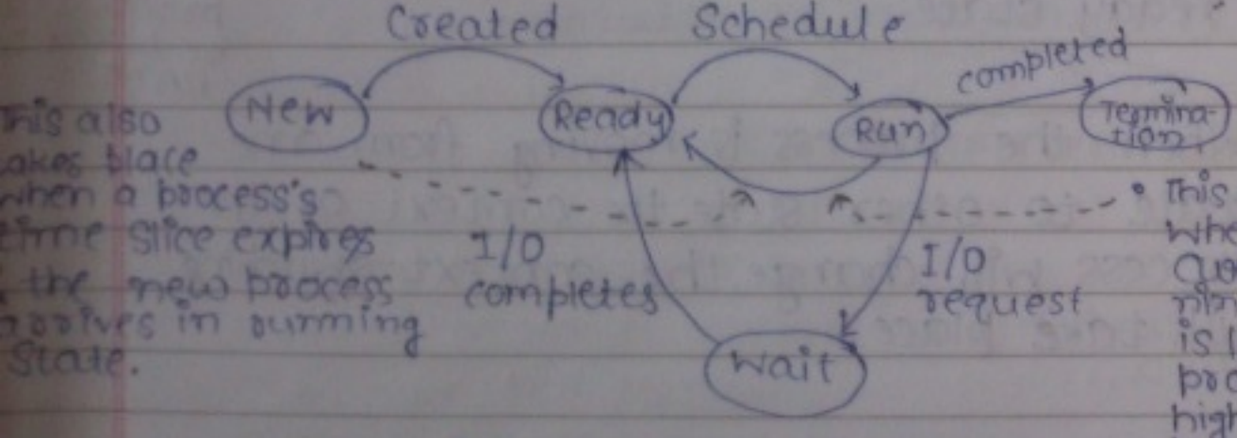
- If the running process completes process moves to termination or completion state.



For Non-preemptive

- In wait state also, we can have multiple no. of processes.

Multiprogramming OS → non-pre-emptive
→ pre-emptive [or Multitasking or Timesharing]



This also takes place when a process's time slice expires & the new process arrives in running state.

This takes place when the currently running process is low priority process & a high priority process arrive the low priority pro is preempted.

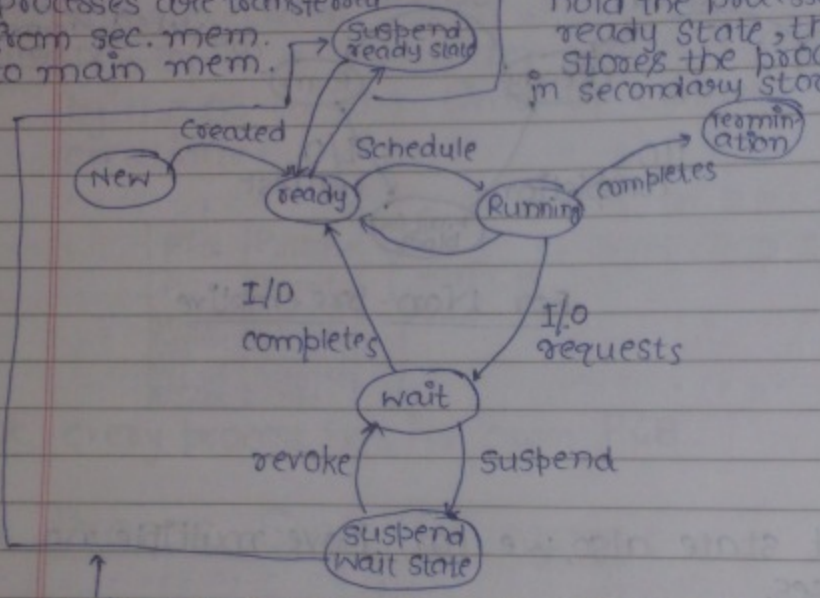
For Pre-emptive

Cosmos

* When the process is in the ready, running or wait state, it means process is in the main memory.

* & When the comp. or mem. has sufficient resources, then the suspended processes are transferred from sec. mem. to main mem.

When the main memory is not sufficient to hold the processes in ready state, the OS stores the processes in secondary storage *

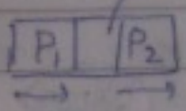


this takes place when the process has completed I/O, but immediately goes to suspend wait state due to lack of resources, but we can't place it into suspend wait state because it has already completed its I/O operation & must be placed into suspend ready state.

lack of resources in this context means that the RAM is full, & process can be placed in RAM.

- When the process is moving from one state to other state, the context of the process will change, the context-switching will take place.

Cosmos



Saving the context of one process & loading the context of other process is called as context switching

★ context-switching takes place when we have atleast two processes.

★ If the context of the process is more, the context switching time also increases which is undesirable.

★ When there is one process only, though states of the process changes, so context changes, but this is not context-switching.

Processes w.r.t. their execution time

CPU bound processes

- The processes which require more CPU time are called CPU bound processes.
- These processes will spend more time in running state.

I/O bound processes

- The processes which require more I/O time are called as I/O bound processes.
- These processes will spend more time in wait state.

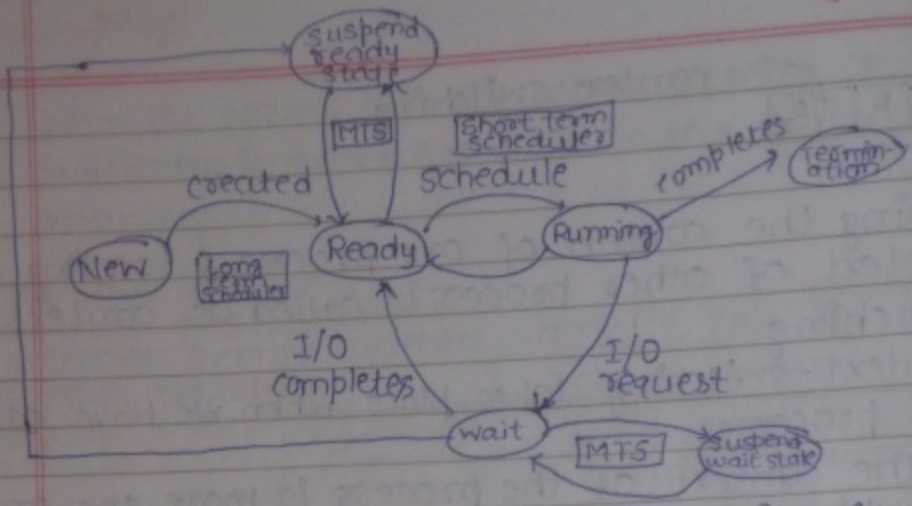
★ In OS, we have 3-Schedulers:-

1. Long-term Scheduler or Job Scheduler
2. Short-term " " CPU "
3. Mid-term " or Medium-term Scheduler.

→ Long-Term Scheduler :-

It is responsible of creating & bringing new processes into the system.

MTS:- mid-term scheduler



→ Short-Term Scheduler :- is responsible for selecting one of the process from ready state to scheduling onto running state.

→ Mid-Term Scheduler :- is responsible for suspending & resuming the processes. The job done by mid-term scheduler is called as swapping.

→ Dispatcher :- is responsible for saving the context of one process & loading the context of another process. Therefore, the context switching is done by the dispatcher.

★ If all (or majority) of the processes are I/O bound processes, then running state becomes empty & wait queue increases & hence throughput decreases.

★ If all (or majority) of the processes are CPU bound processes, then each process will take more time in running state, CPU utilization increases, but no. of processes executed per second decreases & hence throughput decreases.

Cosmos

★ Long term Scheduler should select good combination of CPU bound or I/O bound processes in order to get good throughput of the system.

→ Degree of multiprogramming :- the no. of jobs present in the mem. at any pt. of time is called as degree of multiprogramming.
The Long-Term Scheduler controls degree of multiprogramming.

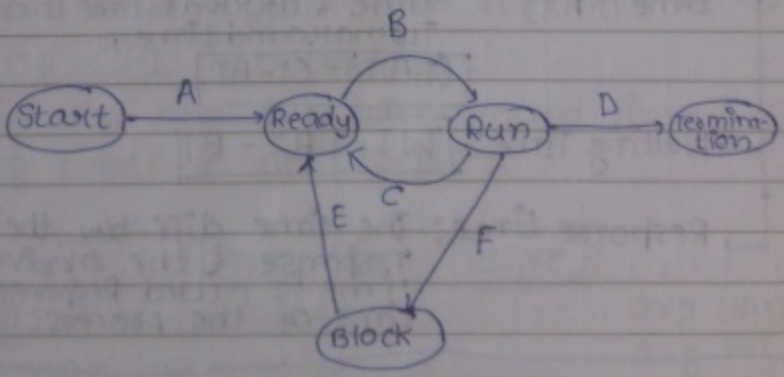
Q. Consider a system with 'N' CPU processors, then what is the min. & max. no. of processes that may be present in ready, running & block states

Ans.

Running	Ready	Block
Min. → 0	Min. → 0	Min. → 0
Max. → N	Max. → any no. of processes	Max. → any no. of processes. (depends upon the size of main mem

Min. is 0 because LTS (Long-Term Scheduler) is not creating any process.

Q.



Consider the statements below :-

1. If a process makes a transition D, it would result in another process making a transition A immediately.

2. A process P_2 in block state can make a transition E while another process P_1 is in the running state.
3. The OS uses pre-emptive scheduling.
4. The OS uses non-pre-emptive scheduling.

Which are true?

Ans. (2) & (3)

(1) is not true because there is no relation b/w A & D.

process

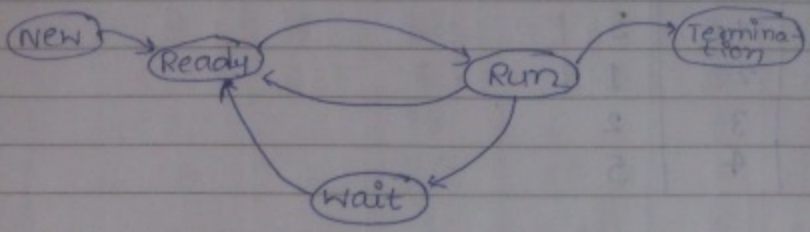
- **Arrival time (AT):** The time when the process arrives in the ready state is called arrival time of the process.
- **Completion time (CT):** The time when the process completes its execution is called completion time of the process.
- **Burst time (BT):** The time required for the process for its execution is called Burst time of the process. (BT = CPU time + I/O time)
  - no waiting time.
- **Turnaround time (T.A.T):** The time diff. b/w completion time & arrival time is called Turnaround time.

$$T.A.T = CT - AT$$

- **Waiting Time:** $WT = T.A.T - BT$
- **Response time:** The time diff. b/w the 1st response & the arrival time is called Response time of the process.

Cosmos

CPU Scheduling



The short-term scheduler will use CPU scheduling algorithm in ready state.

hen we — ① When process moves from running to termination
ll use
scheduling: (i.e. when process completes)

② When process is moving from running to wait state (i.e. when process requires I/O operation.)

③ When the process is moving from running to ready (i.e. when time slice expires or the low priority is pre-empted by high priority process.)

④ When the process moves from New to ready. (This takes place because the new process can be a high priority process)
[We have to apply CPU scheduling every time a new process is created.]

⑤ When the process moves from wait to ready state (because of similar reason of pt. 4).

- **Goal of CPU Scheduling:-**
To minimize the avg. turnaround time & avg. waiting time of the processes.

① FCFS (First Come First Serve)

- **Criteria:-** Arrival Time.
- **Mode:-** non-preemptive

avg TAT = ?
avg WT = ?

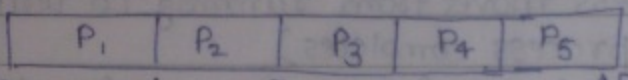
** As criteria is arrival time, when two processes with same arrival time, then schedu
Q. that process with lowest process id.

Cosmos

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

Avg. TAT =

Gantt Chart



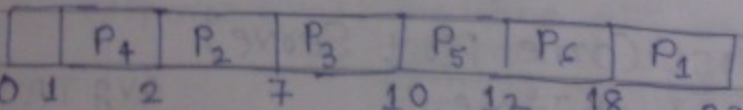
$$\text{avg TAT} = \frac{4+6+6+7+11}{5} = \frac{34}{5} = 6.8$$

$$\text{avg WT} = \frac{0+3+5+5+6}{5} = \frac{19}{5} = 3.8$$

Q.

P. No.	A.T.	B.T.	C.T.	TAT	W.T. → TAT-BT
1	6	4	22	16	12
2	2	5	7	5	0
3	3	3	10	7	4
4	1	1	2	1	0
5	4	2	12	8	6
6	5	6	18	13	7

→ C.T. - A.T.

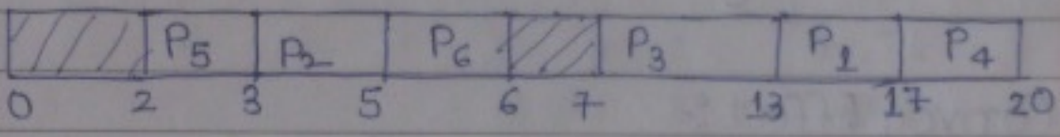


$$\text{Avg. TAT} = \frac{50}{6} = 8.33$$

$$\text{Avg. WT} = \frac{29}{6} = 4.833$$

Q.

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	8	4	17	9	5
2	3	2	5	2	0
3	7	6	13	6	0
4	10	3	20	10	7
5	2	1	3	1	0
6	3	1	6	3	2



Avg. TAT = $\frac{31}{6}$ = 5.16

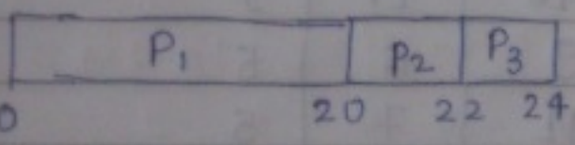
Avg. WT = $\frac{14}{6}$ = 2.3

Note:-

** If the arrival times of the processes are matching then schedule the process with lowest process id.

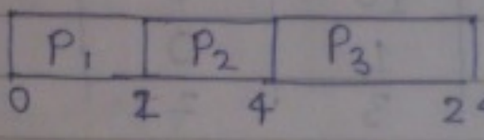
Q.

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	0	20	20	20	0
2	1	2	22	21	19
3	2	2	24	22	20



avg. WT. = $\frac{19+20}{3}$ = 13.

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	0	2	2	2	0
2	1	2	4	3	1
3	2	20	24	22	2



Avg. WT. = $\frac{3}{3} = 1$

★★ Convoy Effect :-
In FCFS, if the 1st process is having a large burst time, then it will have a huge impact on the avg. waiting time of all the remaining process. So the effect is called as the convoy effect.

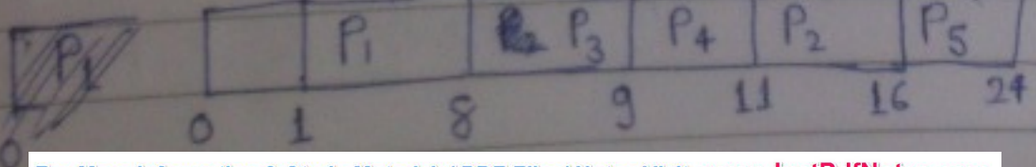
Shortest Job First

Criteria :- Burst Time
Mode :- Non-preemptive & Pre-emptive both

Non-Preemptive :-

Q.

P.No.	AT	BT	C.T.	T.A.T.	W.T.
1	1 ✓	7	8	7	0
2	2 ✓	5	16	14	9
3	3 ✓	1	9	6	5
4	4 ✓	2	11	7	5
5	5 ✓	8	24	19	11



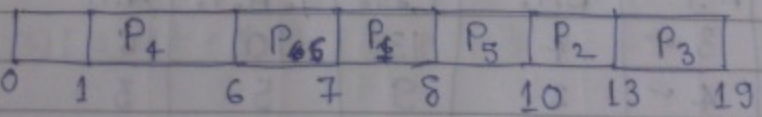
Avg. T.A.T. = $\frac{53}{5}$ = 10.6

Avg. W.T. = $\frac{30}{5}$ = 6

Note:- If the burst times of the processes are matc
** then schedule the process with lowest arrival
time.

Q.

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	6	1	7	1	1
2	3	3	13	10	7
3	4	6	19	15	9
4	1	5	6	5	0
5	2	2	10	8	6
6	5	1	8	3	2



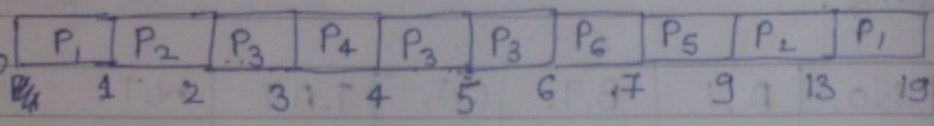
Avg. TAT = $\frac{42}{6}$ = 7

Avg. WT = $\frac{24}{6}$ = 4

Q.

Pre emptive

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	0	~~7~~ 6	19	19	12
2	1	~~5~~ 4	13	12	7
3	2 ✓	~~3 2~~ 1	6	4	1
4	3 ✓	1	4	1	0
5	4 ✓	2	9	5	3
6	5 ✓	1	7	2	1

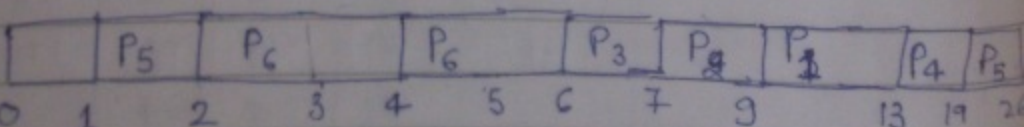


Avg. T.A.T = $\frac{43}{6} = 7.16$

Avg. WT. = $\frac{24}{6} = 4$

Q.

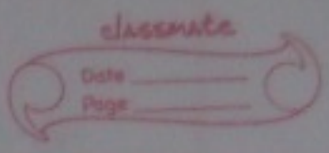
P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	3 ✓	4	13	10	6
2	4 ✓	2	9	5	3
3	5 ✓	1	7	2	1
4	2	6	19	17	11
5	1	~~8~~ 7	26	25	17
6	2 ✓	~~4~~ 2 1	6	4	0



Avg. T.A.T = $\frac{63}{6} = 10.5$

Avg. WT = $\frac{38}{6} = 6.33$

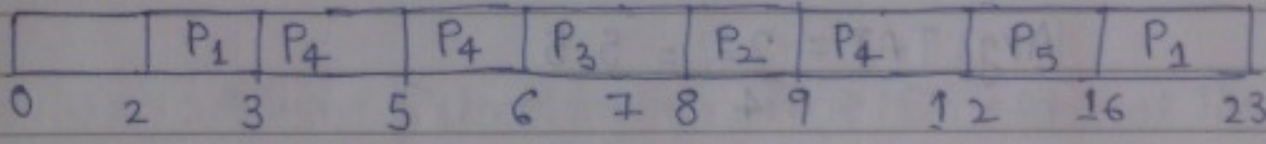
Cosmos



★★ If we have same burst time, then we will check the ~~burst~~ arrival time, if arrival times are also same, then choose process with lowest process id.

Q.

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	2	~~8~~ 7	23	21	13
2	7 ✓	1	9	2	1
3	6 ✓	~~2~~ 1	8	2	0
4	3 ✓	~~6~~ ~~4~~ 3	12	9	3
5	5	4	16	11	7

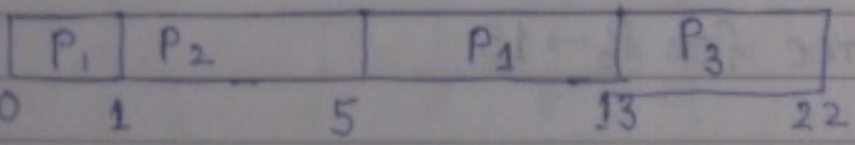


Avg. TAT $= \frac{45}{5} = 9$

Avg. WT $= \frac{24}{5} = 4.8$

Q.

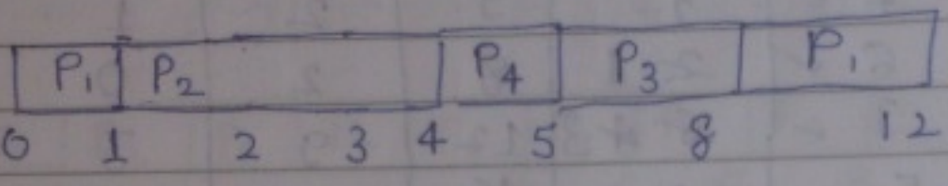
P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	0	~~9~~ 8	13	13	~~5~~ 4
2	1	4	5	4	0
3	2	9	22	20	11



Avg. WT $= \frac{15}{3} = 5$

Q.

P.No.	AT	BT	C.T.	T.A.T.	W.T.
1	0	~~5~~ 4	12	12	7
2	1 ✓	3 ~~2~~ 1	4	3	0
3	2	3	8	6	3
4	4 ✓	1	5	1	0

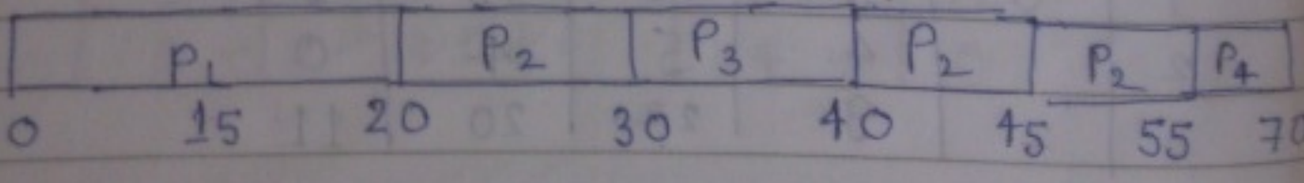


Avg. WT. = $\frac{10}{4}$ = 2.5

Avg. TAT = $\frac{22}{4}$ = 5.5

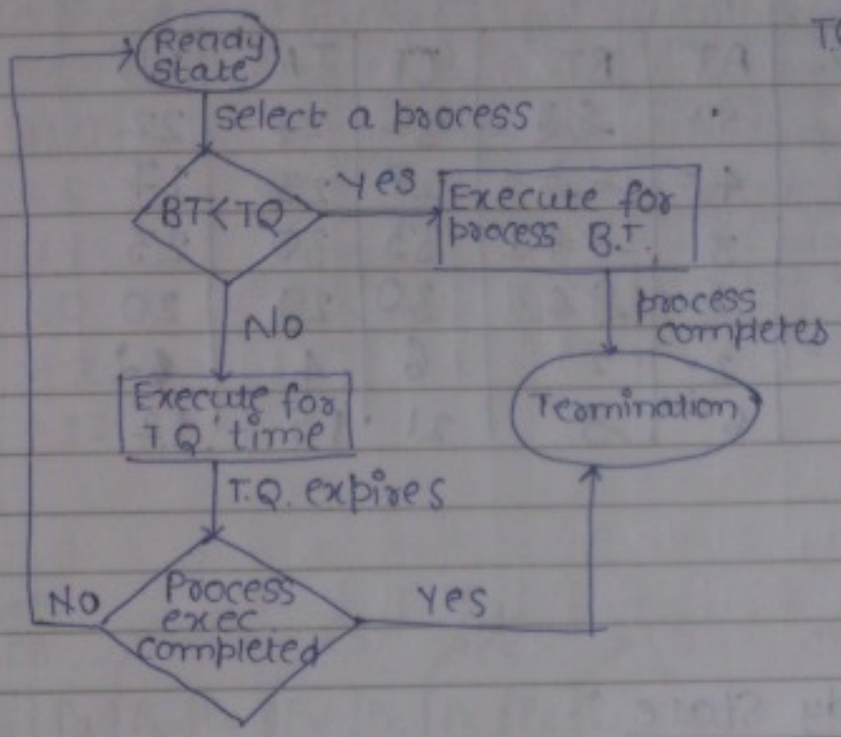
Q.

P. No.	AT	BT	C.T.	T.A.T.	W.T.
1	0 ✓	~~20~~ 5	20	20	0
2	15 ✓	~~25~~ ~~15~~ ~~10~~ 5	~~55~~ 5	~~40~~ 0	~~25~~ ~~10~~ 15
3	30 ✓	10	40	10	0
4	45	15	70	25	10



Waiting Time for $P_2 \rightarrow 15$

Round-Robin Scheduling:



T.Q → Time Quantum or Time slice

Criteria: Time Quantum or Time Slice, Arrival Time

Mode: preemptive

★★ When arrival time of the processes are same, we schedu that process which has lower pid.

Q. T.Q. = 2

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

P_1	P_2	P_3	P_1	P_4	P_5	P_2	P_6	P_5	P_2	P_6	P_5
0	2	4	6	8	9	11	13	15	17	18	19

we write P_1 here because P_1 doesn't gets completed at t=2. we write P_1 after P_2 & P_3

Ready State :-

P_1, P_2, P_3, P_1, P_4, P_5, P_2, P_6, P_5, P_2, P_6, P_5

we wrote this as it is the only process at t=0

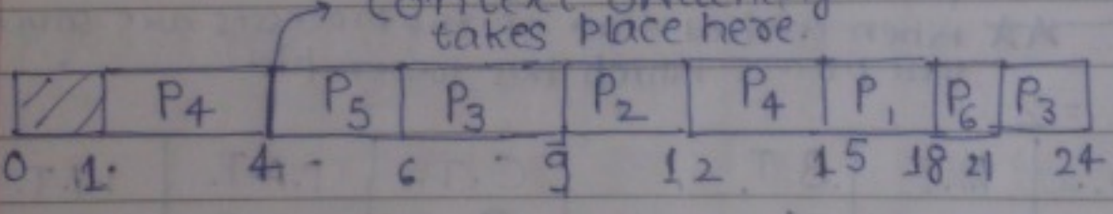
after t=2 P_1 is executed for 2 sec we write P_2 & P_3 at t=2 P_2 & P_3 arrive

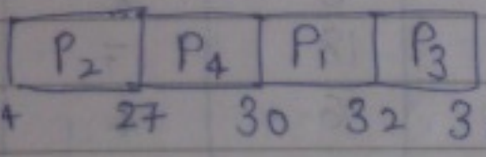
Q. TQ = 3

P.No.	AT	BT	CT	TAT	WT
1	5	~~5 2~~	32	27	22
2	4	~~6~~ 3	27	23	17
3	3	~~7 4~~ 1	33	30	23
4	1	~~9 6~~ 3	30	29	20
5	2	2	6	4	~~4~~ 2
6	6	~~3~~	21	15	12

Ready State :-

$P_4\ P_5\ P_3\ P_2\ P_4\ P_1\ P_6\ P_3\ P_2\ P_4\ P_1\ P_3$





Avg TAT = $\frac{128}{6} = 21.3$

Avg WT = $\frac{96}{6} = 16$

★ Response Time :-

Time diff. b/w 1st response & arrival time.

Respons

Process	Response Time (1st response - AT)
P_1	15 - 5 = 10
P_2	9 - 4 = 5
P_3	6 - 3 = 3
P_4	1 - 1 = 0
P_5	4 - 2 = 2
P_6	18 - 6 = 12

Ready State explanation:-

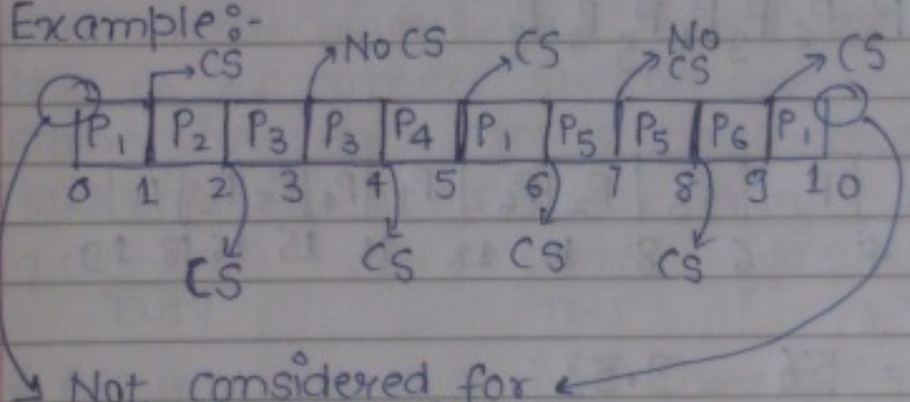
① The process P_4 arrives at t=1, & its the 1st process, so we put it in ready queue & execute it for 3 sec (i.e from t=1 to t=4).

② From t=1 to t=4, 3 new processes arrive in order P_5 P_3 & P_2, so we place P_5, P_3 & P_2 in the ready queue, now process P_4 is not complete so we put it after P_2, so now ready queue contain $P_4 P_5 P_3 P_2 P_4$

③ Now, we will execute P_5 from t=4 to t=6 (as BT = 2) & it gets completed, so we don't put it in ready queue (at t=6 P_1 & P_6 also arrives, so we put in queue P_1 & P_6)

④ Now, we execute P_3 from t=6 to t=9 [but P_3 doesn't gets completed], so we put P_3 back in the queue. queue is $P_4 P_5 P_3 P_2 P_4 P_1 P_6 P_3$ & so on.

Example:-



CS:- Context Switching

★ In Round Robin if TQ is less, then no. of context switches will increases & response time decreases. if TQ is more, no. of context switches decreases & response time increases.

★ If TQ is very high, then Round-Robin degenrates into FCFS.

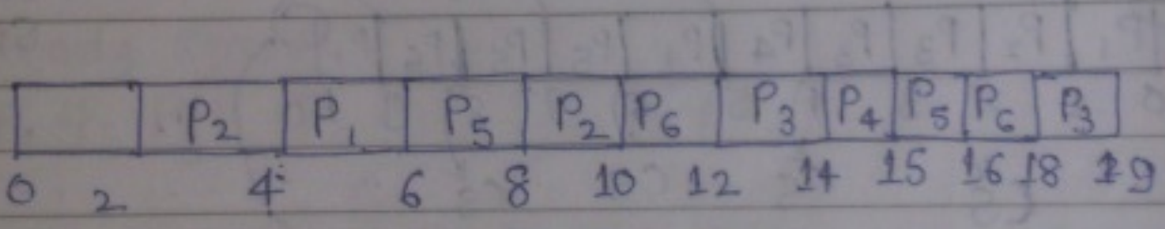
TQ = 2

Q.

P.No.	A.T.	B.T.	CT	TAT	WT	RT
1	3	2	6	3	1	1
2	2	~~4~~ 2	10	8	4	0
3	6	~~8~~ 7	~~19~~ 19	~~13~~ 13	11	6
4	8	1	15	7	6	6
5	4	~~3~~ 1	16	12	9	2
6	5	~~4~~ 2	18	13	9	5

ready state :-

$P_2\ P_1\ P_5\ P_2\ P_6\ P_3\ P_4\ P_5\ P_6\ P_3$



Avg. TAT = $\frac{57}{6} = 9.33$

Avg. WT = $\frac{40}{6} = 6.6$

Avg. response time = $\frac{20}{6} = 3.33$

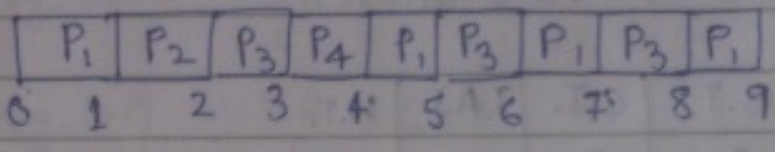
Consider a system with 4 jobs P_1, P_2, P_3, P_4 arriving in ready ~~reque~~ queue in same order at t=0. The B.T. respec. are 4,1,8,1. Then what is the C.T. of P_1 assuming round-robin scheduling with TQ = 1?

Cosmos

Ready State:-

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

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



Q. Consider 'n' processes sharing the CPU in roun robin fashion. If the context switching time is 's' u then what must be the Time Quantum "q" such th the no. of context switches are reduced, but at th same time each process is guaranteed to get its turn/job at the CPU for every 't' sec. of time

(a) $q = \frac{t - ns}{n+1}$ (b) $q = \frac{t + ns}{n-1}$ (c) $q = \frac{t - ns}{n-1}$ (d) $q = \frac{t -$ $n+$

Ans. ★ Assume the arrival time of all processes to be 0 & Burst time to be very large.

s

P_1 P_2 P_3 P_4 P_1

t

ns → time for context switches

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

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

context switch → ns

P_1 P_2 P_3 P_4 P_1

$q(n-1)$ t

[(n-1) time quantums.]

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

Date

17.06.12

Cosmos

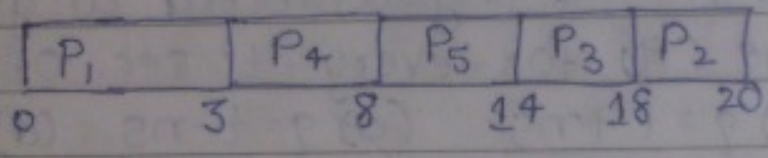
Longest Job First

Criteria:- Burst Time

Mode:- Non-Preemptive & Pre-emptive

Q. (Non-preemptive)

P.No.	A.T.	B.T.	C.T.	T.A.T	W.T.
1	0	3	3	3	0
2	1	2	20	19	17
3	2	4	18	16	12
4	3	5	8	5	0
5	4	6	14	10	4



Avg TAT = $\frac{43}{5}$ = 10.6

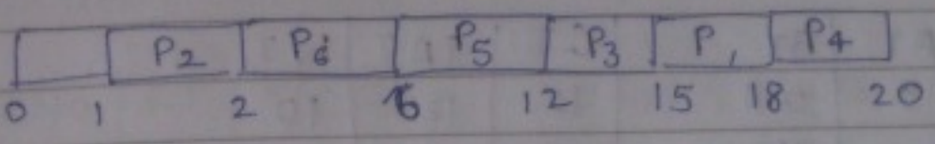
Avg WT = $\frac{33}{5}$ = 6.6

Q.

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
1	3	3	18	15	12
2	1	1	2	1	0
3	2	3	15	13	10
4	4	2	20	16	14
5	6	6	12	6	0
6	2	4	6	4	0

Note:- The B.T. of the processes are matching, then schedule the process having lowest arrival time.

Cosmos



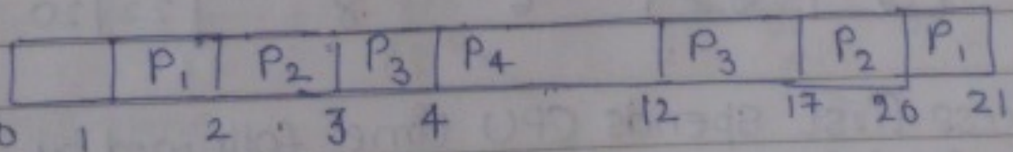
Avg. TAT = $\frac{55}{6}$ = ~~9.16~~ 9.1

Avg. WT = $\frac{36}{6}$ = 6

Pre-emptive

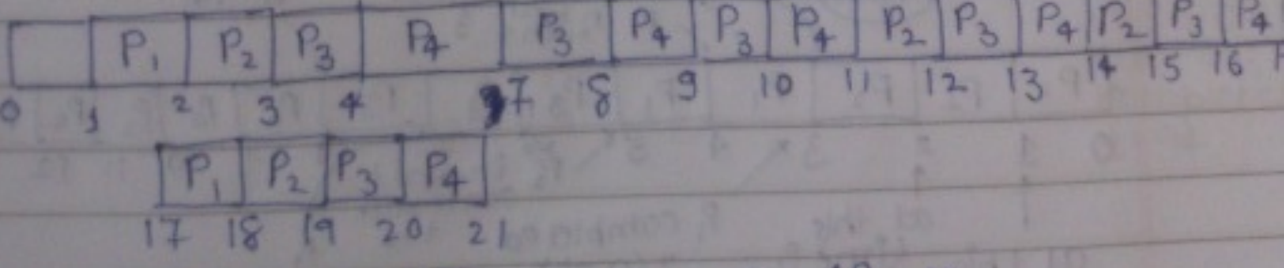
(2)

P.No.	AT	BT	CT.	T.A.T.	W.T.	C.T	TAT	W.T.
1	1	21	21	20	18	18	17	15
2	2	~~4~~ 3 2	20	18	14	19	17	13
3	3 ✓	~~6~~ 5 4	17	14	8	20	17	11
4	4 ✓	~~8~~ 5 4 3 2 1	12	8	0	21	17	9



ty
nswer:- Avg. TAT = $\frac{60}{4}$ = 15

Avg. WT = $\frac{40}{4}$ = 10

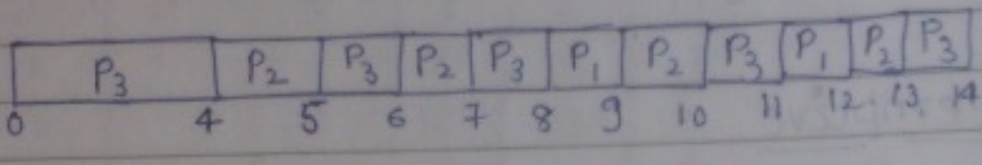


Correct Answer:- Avg. TAT = 17 Avg. WT = $\frac{48}{4}$ = 12

Cosmos

Q.

P.No.	AT	BT	C.T.	T.A.T.	W.T.
1	0	2 1	12	12	10
2	0	4 3 2 1	13	13	9
3	0	8 4 3 2 1	14	14	6



Avg TAT = 13

Avg. WT = $\frac{25}{3}$ = 8.33

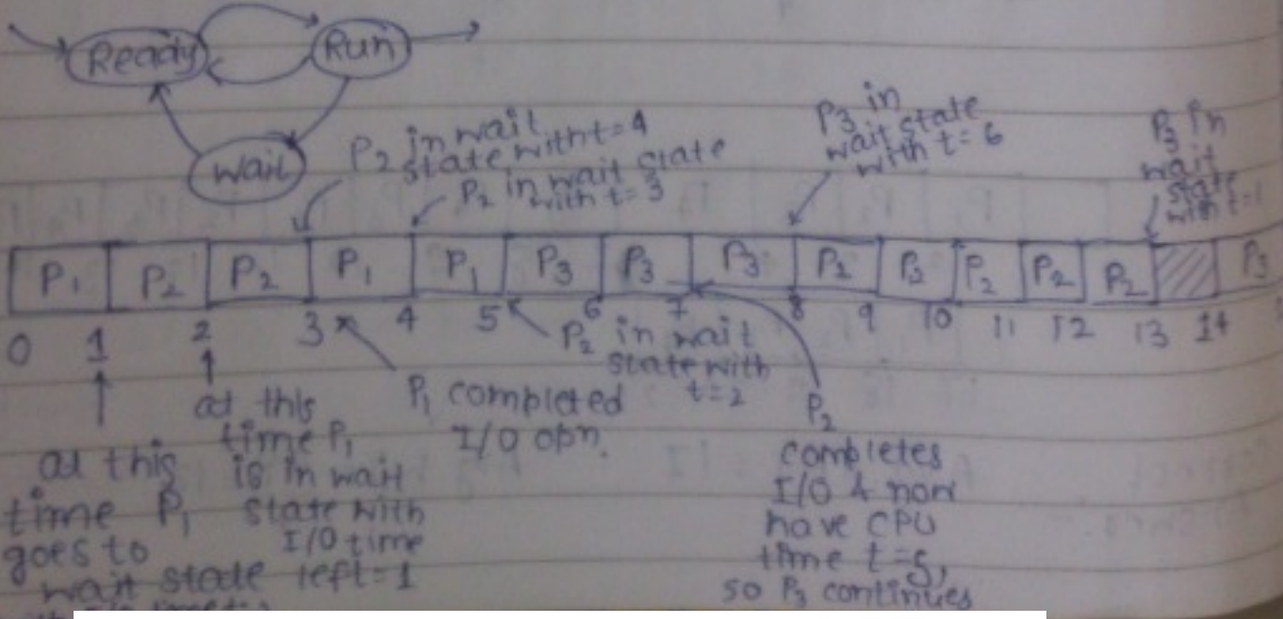
Preemptive Shortest Job First :-

Q

P.No.	AT	Execution Time: CPU Time	I/O Time	CPU Time	C.T.	T.A.T.
1	0	1 0	2 1 0		5	5
2	1	2 1 0	4	5	13	12
3	2	3 2 1	6	8	22	20

Note :- ① Process first spends CPU time followed by I/O time & followed by CPU time again.

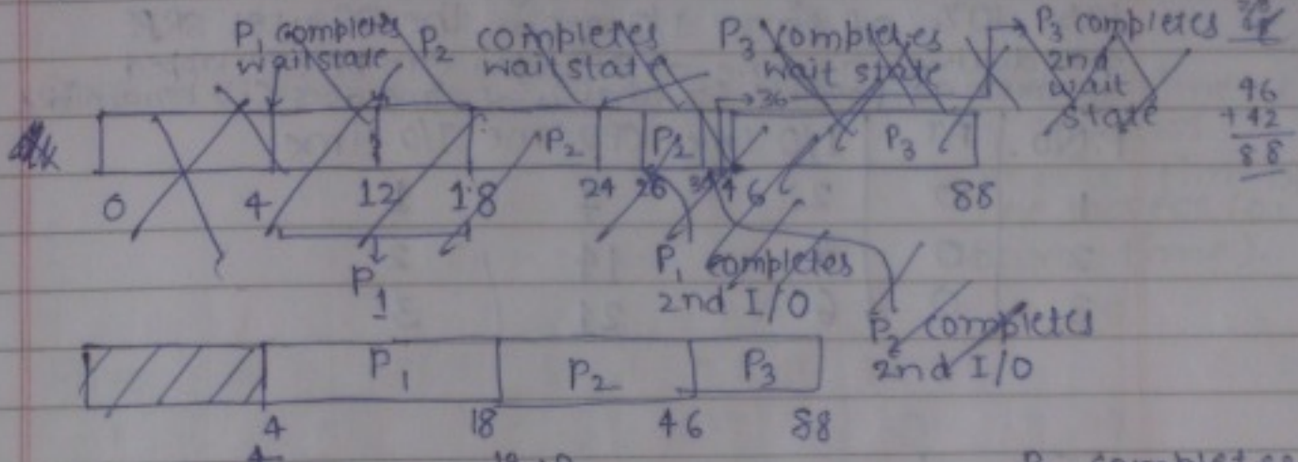
② I/O of the processes can be overlapped as much as possible.

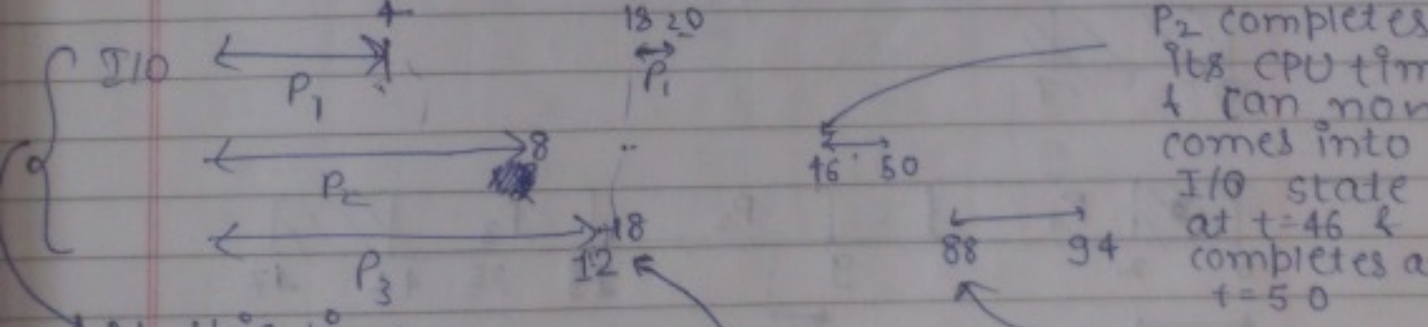


★ We don't calculate T.A.T. & W.T. in these questions.

Q.

P.No.	AT	I/O Time	CPU time	I/O time	C.T.
1	0	40	10	20	20
2	0	80	28	4	50
3	0	120	42	6	94





At this time all the I/O's can be overlapped & hence, P_1 I/O time, P_2 I/O time, P_3 I/O time overlaps.

P_1 completes CPU time at t=18 & its I/O time can start at t=18 & completes at t=20

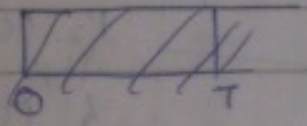
P_2 completes its CPU time & can now comes into I/O state at t=46 & completes at t=50

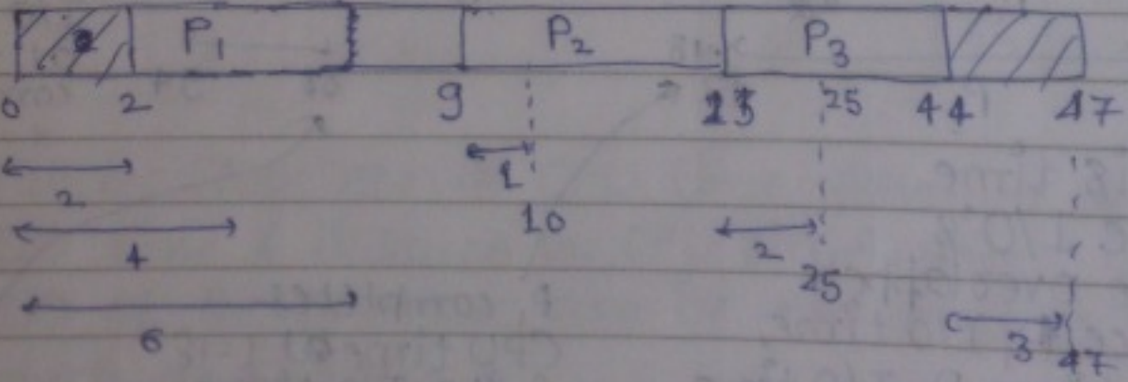
P_3 completes its CPU time at 88 & is ready for its I/O operation at t=88 & completes at t=94.

Cosmos

Q. Consider 3 processes all arriving at time t=0 with total execution time of 10,20,30 time units respec. Each process spends the 1st 20% of execution time doing I/O, the next 70% of time doing computation & last 10% of time I/O again, the OS uses SRJF scheduling, assume all I/O can be overlapped as much as possible, for what % of time does CPU remain idle.

P.No.	AT	I/O time	CPU time	I/O time
1	0	2	7	1
2	0	4	14	2
3	0	6	21	3





CPU → % idle $\frac{5}{47} \times 100 = \frac{500}{47} \approx 10.6\%$

$$47 \overline{)500}\ (10.6$$
$$-47$$
$$300$$

Highest Response Ratio Next (HRRN)

It favours the shorter jobs & also limits the waiting time of longer jobs

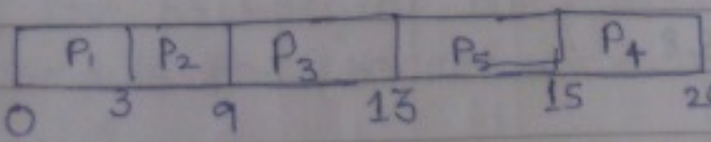
Criteria:- Response Ratio
Mode:- Non-preemptive

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

where WT:- waiting time of the process
S:- service time of the process (or Burst time).

①

P.No.	A.T.	B.T.	C.T.	T.A.T.	W.T.
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	17	5



at t=9:

$RR(P_3) = \frac{(9-4)+4}{4} = \frac{9}{4}$ } → WT = 9 − AT

$RR(P_4) = \frac{(9-6)+5}{5} = \frac{8}{5}$

$RR(P_5) = \frac{(9-8)+2}{2} = \frac{3}{2}$

Avg. TAT $= \frac{40}{5} = 8$

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

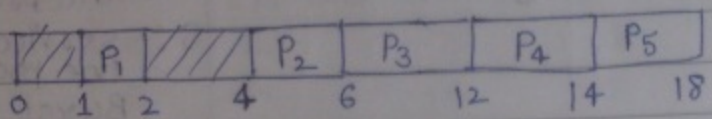
at t=13:

$RR(P_4) = \frac{(13-6)+5}{5} = \frac{12}{5} = 2.4$

$RR(P_5) = \frac{(13-8)+2}{2} = \frac{7}{2} = 3.5$

(2)

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



at t=6 $RR(P_3) = \frac{(6-5)+6}{6} = \frac{7}{6}$

$RR(P_4) = \frac{(6-6)+2}{2} = 1$

at t=12 $RR(P_4) = \frac{(12-6)+2}{2} = \frac{8}{2} = 4$

$RR(P_5) = \frac{(12-7)+4}{4} = \frac{9}{4}$

Avg TAT $= \frac{29}{5} = 5.8$

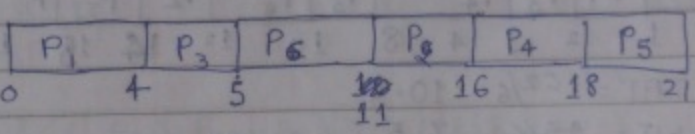
Avg WT $= \frac{14}{5} = 2.8$

Priority Based Scheduling

Criteria :- Priority
Mode :- Non-preemptive

Priority	P.No.	A.T.	B.T.	C.T	T.A.T	W.T.
4	1 ✓	0	4	4	4	0
5	2 ✓	1	5	16	15	10
7	3 ✓	2	1	5	3	2
2	4 ✓	3	2	18	15	13
1	5	4	3	21	17	14
6	6 ✓	5	6	11	6	0

Assumption → Highest no. → Highest priority



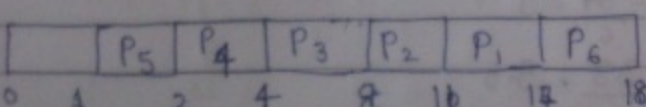
Avg TAT = $\frac{60}{6} = 10$ Avg WT = $\frac{39}{6} = 6.5$

Note :- If the priority of the process are matching, then schedule the process with lowest arrival time.

P.No.	AT	BT	Priority	CT	TAT	WT
1	4	6	4	17	13	7
2	6 ✓	3	5	11	5	2
3	3 ✓	4	6	8	4	0
4	2 ✓	2	6	4	2	0
5	1	1	7	2	1	0
6	2	2	3	19	17	15

Avg TAT = $\frac{70}{6} = 7$

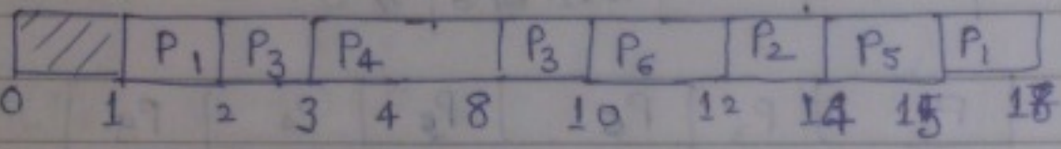
Avg WT = $\frac{21+3}{6} = 4$



Now,

Q Preemptive

P.NO.	AT	BT	Priority	CT	TAT	WT
1	1	~~4~~ 3	4	18	~~16~~ 17	13
2 ✓	2	~~2~~ 1	5	~~13~~	12	10
3 ✓	2	~~3~~ 2	7	10	8	5
4 ✓	3	~~5~~ 4	8	8	5	0
5 ✓	3	1	5	~~15~~	12	11
6 ✓	4	2	6	12	8	6

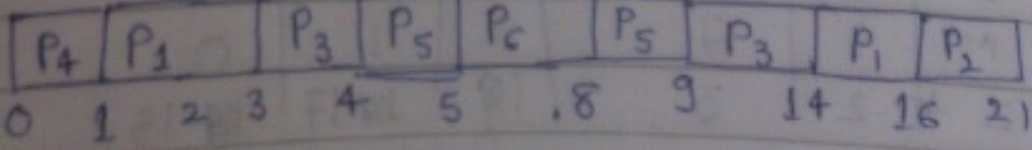


Avg TAT = ~~62~~/6 = 10.3

Avg WT = 45/6 = 7.5

Q.

P.No.	AT	BT	Priority	CT	TAT	WT
1	1	~~4~~ 3	5	16	15	11
2	2	5	2	21	19	14
3	3	~~8~~ 5	6	14	11	5
4	0 ✓	11	4	1	1	0
5	4 ✓	~~2~~ 1	7	9	5	3
6	5 ✓	3	8	8	3	0



Avg TAT = $\frac{54}{6}$ = 9

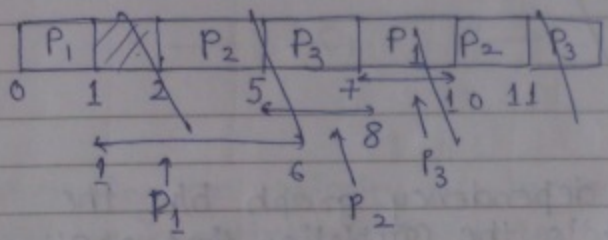
Avg WT = $\frac{33}{6}$ = 5.5

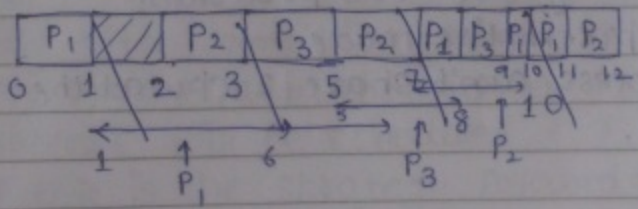
Cosmos

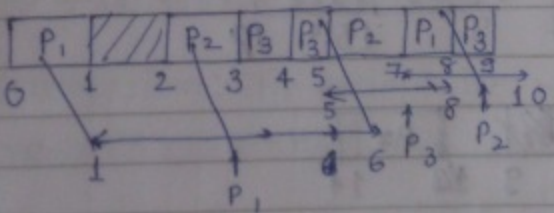
Q.

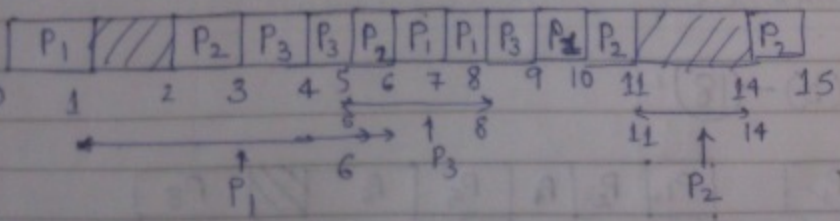
P.No.	A.T.	Priority	CPU time	I/O time	Burst time: CPU time	CT
P_1	0	2	1	5	1	10
P_2	2	3 → lowest	1	3	1	15
P_3	3	1 → highest	2	3	1	9

I/O can be overlapped as much as possible

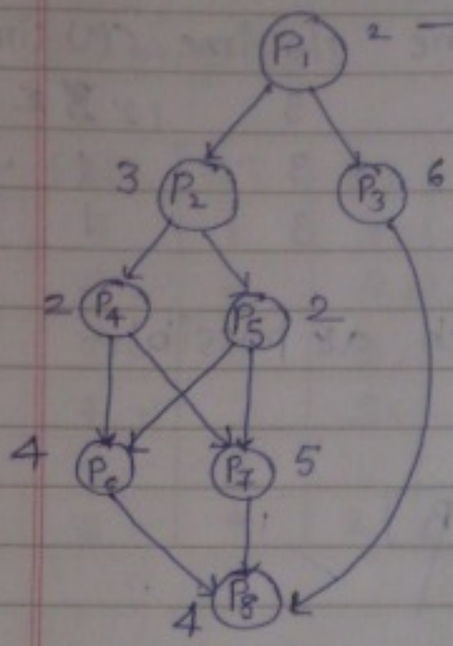








Cosmos



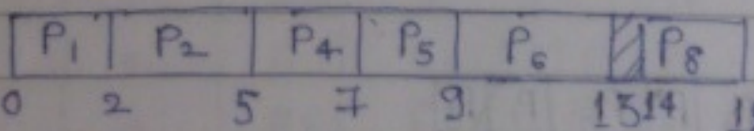
Only after completion of P_1, only then we can start P_2 & P_3

Q. Consider the dependency graph b/w the processes what is the completion time of all the processes using two-CPU processors?

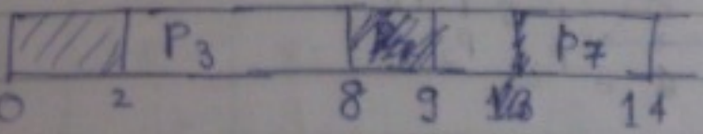
Note:- (1) Use non-preemptive mode
(2) One process can't share 2-cpu at the same time.

ns.

CPU1:-



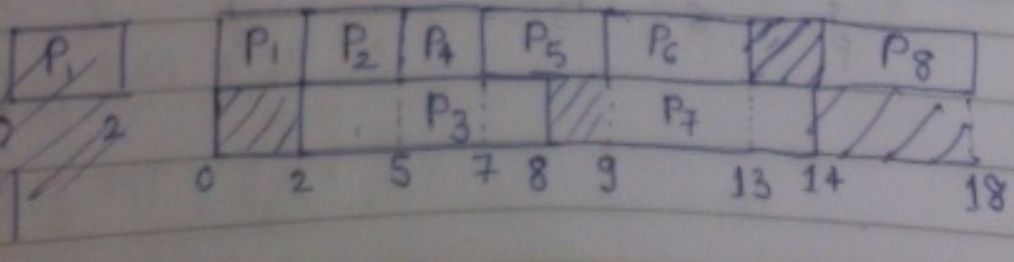
CPU2:-



Processors CT

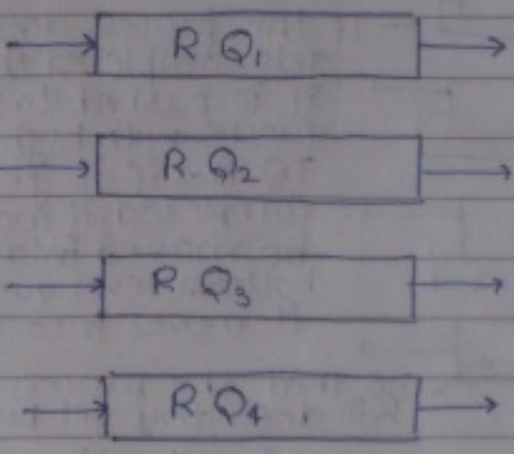
P_4

Ans → (b) – (18)



Cosmos

Multilevel Queue Scheduling



(Distributing processes into multiple ready queues.)

★ Generally high priority processes are placed in top level ready queue

Generally low priority processes are placed in bottom-level ready queue.

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

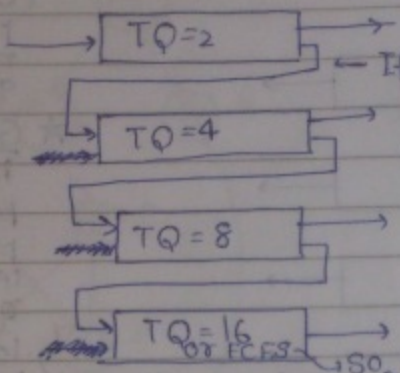
★ Only after completion of all the processes in top-level ready queue, the further level ready queue processes will be scheduled.

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

★ Starvation :- The indefinite waiting of a process is known as starvation.

Multilevel Feedback Queue Scheduling

Cosmos



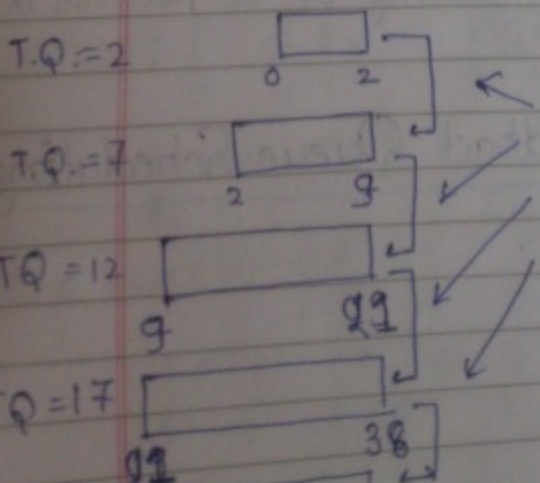
If the process doesn't gets completed then it is pushed to next queue with higher TQ. But if it gets completed, it wont be moved into next bottom-level queue & it moves into termination State.

So that the process will definitely get processed.

It avoids the starvation & at the same time gives pref. to high priority process.

Q. Consider a system having process with BT = 40 s. If the Queue TQ = 2 & at each level the T.Q. increases by 5 time units. How many times the job will be interrupted & on which queue it will terminate?

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



Cosmos

	Algorithm	Starvation
1.	FCFS	No
2.	NP→SJF	Yes
3.	SRTF (Shortest remaining time 1st)	Yes
4.	NP→LJF	Yes
5.	LRTF [P→LJF]	Yes
6.	Round Robin	No
7.	NP priority	Yes
8.	Pre-priority	Yes
9.	HRRN	No
10.	M.L.Q.	Yes
11.	M.L.F.Q.	No

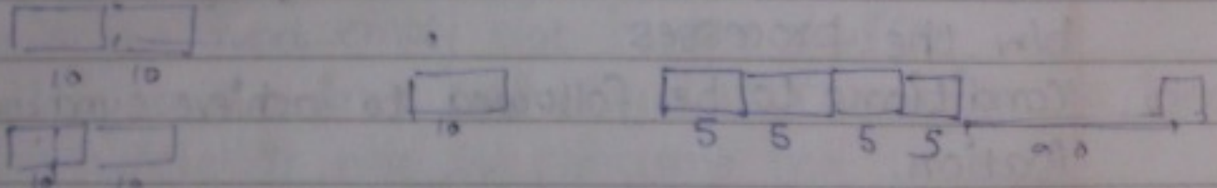
Q. Consider a system having only 2 processes, both of which alternate 10 sec of CPU burst/time with 90 sec. of I/O time. Both the processes were created nearly at the same time. The I/O of both the processes can proceed in parallel. Which of the following scheduling strategy will result in least CPU utilization over a long period of time

(a) FCFS

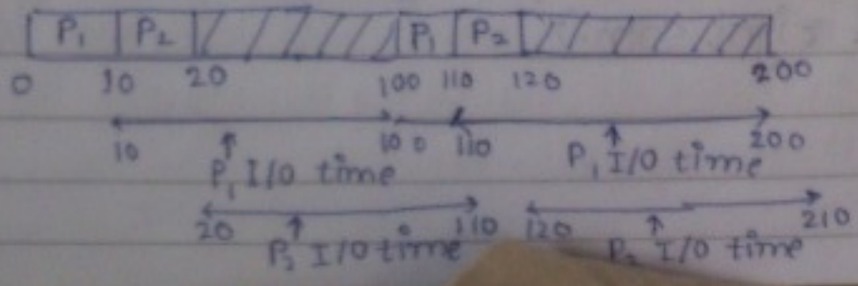
(b) SRTF

(c) Static priority scheduling with diff. priorities of the process

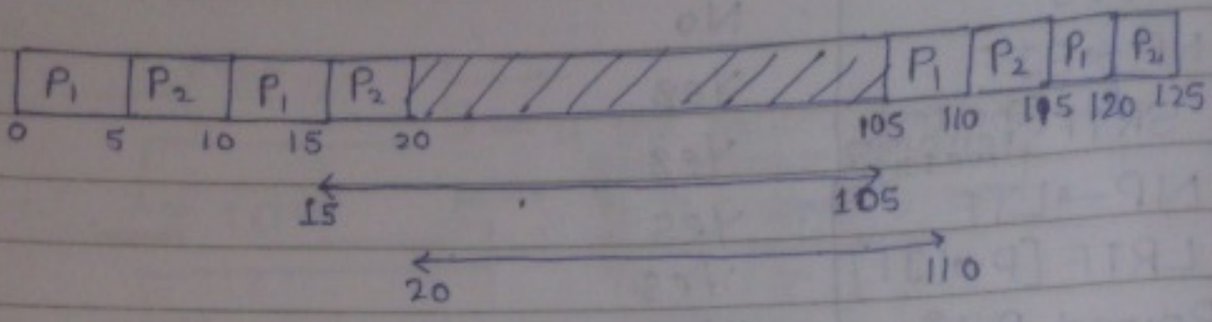
(d) Round Robin with time quantum of 5 sec.



FCFS, SRTF, priority :-



RR :-

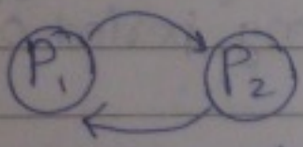


W.T. (for (a), (b), (c)) → 80
W.T. (for (d)) → 85

Synchronisation

Co-operative process :-

The execution of 1 process affected effects the other process or effected by the other process, then those 2 processes are said to be co-operative process, otherwise they are independent process.

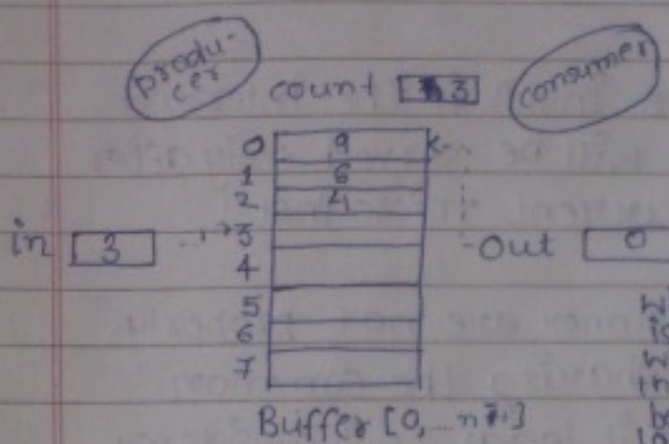


This happens when they have shared variables, shared resources, etc.

1. Problems which arises not having Synchronisation b/w the processes
2. Conditions to be followed to achieve synchronisation.
3. Solutions of for synchronisation. (wrong soln & right soln.)

Cosmos

Producer - Consumer Problem :-




```
int count=0;
void producer(void)
{ int itemp;
  while(true)
  {
    produce_item(itemp);
    while(count==N);
    Buffer[in]=itemp;
    in=(in+1)mod N;
    count=count+1;
  }
```

When buffer is full, i.e. when count=N, then this becomes infinite loop & produ next 3 lines are not executed, so producer can't produce any item till count remains N.

I. load R_P ← M[count]
II. INCR R_P
III. M[count] ← R_P

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

When buffer is empty, then next lines are not executed, & hence consumer wont be able to consume anything.

I. R_E ← M[count]
II. Decr R_E
III. M[count] ← R_E

3 independent micro instn.

★ 'in' is a variable used by the producer in order to identify the next empty slot in the buffer.

★ 'out' is a variable used by the consumer to identify from where it has to consume the item.

★ 'count' is a variable used by the producer & consumer to identify the no. of filled slots in the buffer at any pt. of time.

Cosmos

Shared Resources:-
1. Buffer
2. Count

★ While executing the instⁿ if the interrupt comes, the interrupt will be served only after completion of the current micro instⁿ.

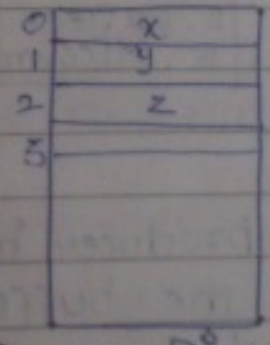
★ The producer & consumer are not properly synchronised while sharing the common variable count, hence it leads to inconsistency.

Ri:- respec process reg.

Printer - Spooler Daemon

(every process will have its own register)

Enter-File :-
1. load Ri, M[in] → loading the Ri with the memory address of next empty slot in the spooler directory.
2. Store SD[Ri], "F-N" → File name
3. INCR Ri
4. Store M[in], Ri



Spooler-Directory [SD]

[This spooler-directory contains all the documents to be printed.]

★ 'in' is a variable used by all the processes in order to identify the next empty slot in the spooler directory.

Shared Resources :-

1. In variable
2. Spooler-Directory

★ out is used by printer only to identify from which slot it has to print the document.

Analysis :-

Ready Queue [P1 | P2]

P1 is trying to print abc.doc

P2 is trying to print xyz.doc

→ Assume P_1 is executing 1st instn.

R_1 [3] M[in] = 3

→ Now P_1 is executing 2nd instn.

SD[R_1] = SD[3]

SD[3] ← abc.doc

→ Now P_1 is executing 3rd instn.

R_1 ← R_1 + 1

R_1 [4]

→ Now, assume P_1 is preempted by P_2 at this stage

→ Now P_2 executes 1st instn.

R_2 ← M[in] M[in] = 3

R_2 ← 3 R_2 [3]

→ Now P_2 executes 2nd instn

SD[R_2] = SD[3]

SD[R_2] ← SD[3] ← xyz.doc [abc.doc is overwritten by xyz.doc

∴ So, loss of data occurs.

The processes are not properly synchronised while sharing common variable in, hence loss of data occurs.

Cosmos

★ It is also possible that Deadlock will occur if processes are not properly synchronised.

Definitions:-

1. Critical Section (CS):-
The portion of program text where the shared variables or shared resources or shared data will be placed.
e.g.
count = count + 1]→ critical section
count = count - 1]→ critical section

2. Non-Critical Section:-
The portion of program text where the independent code of the process will be placed.
e.g.
in = (in+1) Mod N;]→ non-critical section

3. Race Condition:-
The final O/P of any variable depends on execution sequence of processes, so this is called as race-condition.

Condⁿ to be followed to achieve Synchronisation:-

① Mutual Exclusion:-
→ No 2 processes may be simultaneously present inside the critical section at any pt. of time.

Cosmos

→ Only 1 process is allowed in critical section at any pt. of time.

② Progress :- No process running outside the critical section should block the other interested process from entering into critical section when critical section is free.
So, any process running in critical section will block other process trying to enter the critical section.

③ Bounded Wait :- No process should have to wait forever from entering into critical section.
There should be a bound on getting chance for entering into the critical section.
If bounded waiting is not satisfied, there is a possibility of starvation. [if there is no bound, then it can lead to starvation]

④ No assumptions related to h/w or the processor's speed. [soln. is irrespective of h/w or processor's speed]

Solutions :-

① Software type :-
(a) lock variables [No mutual exclusion, no progress]
(b) Strict alteration or Decker's algo [Mutual exclusion, no progress]
(c) Pieterson soln. [Mutual exclusion, progress, bounded waiting]

② Hardware type :-
(a) TSL (Test & Set Lock) instⁿ set

③ OS type :-
(a) Counting semaphore
(b) binary semaphore

④ Compiler & Programming Support Type
(a) Monitors

Lock Variables

Entry-Section
1. Load R_i, M[lock]
2. CMP R_i, #0
3. JNZ to step 1
4. Store M[lock], #1
5. [CS]
6. Store M[lock], #0

Exit-section or remainder section.

Lock
0 — CS is free
1 — CS is busy.

R_i → respec. process register (every process will have its own register).

Analysis :-

Ready Queue :- [P_1 | P_2]

→ I assume lock = 0
→ P_1 executing 1st inst^n
$R_1 \leftarrow$ M[lock]
$\therefore R_1 \leftarrow 0$ R_1 [0]
→ P_1 executing 2nd inst^n
CMP R_1, 0
ZF = 1
→ P_1 executing 3rd inst^n
since ZF = 1, $\therefore$ Program Counter = 4
→ At this stage, P_2 preempts P_1
→ Now P_2 executes 1st inst^n
$R_2 \leftarrow$ M[lock]
$R_2 \leftarrow 0$ R_2 [0]
→ Now P_2 executes 2nd inst^n
CMP R_2, 0
$\therefore$ ZF = 1
→ Now P_2 executes 3rd inst^n which makes PC = 4
→ Now P_2 executes 4th inst^n

Cosmos

→ M[lock] ← 1

[1] lock

→ Now P_2 is executing 5th inst^n [Critical Section]

→ Now if P_1 comes back to execution starting from inst^n 4

M[lock] ← 1

[1] lock

→ Now P_1 executes 5th inst^n & hence enters critical section. (So, both P_1 & P_2 are in critical section).

★ We have proved that both P_1 & P_2 are simultaneous present in the critical section at same time & hence Mutual exclusion is not satisfied, so soln. is bound to be incorrect.

→ If the other process preempts just after last inst^n of critical section & before 6th inst^n, then no other process wont be able to enter the critical section. [when some process pre-empts a process which is executing its last inst^n of CS, i.e. just after CS & just before 6th inst^n, then it wont be able to enter the critical section because the lock is 1 & is not reset to 0]

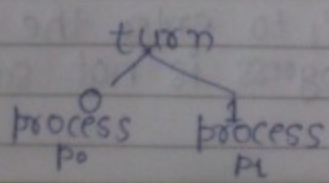
Strict alteration or Dacker's Algo :-

process p_0 code:-

```
while (true)
  { Non_CS();
    while (turn != 0);
    [CS]
    turn = 1;
  }
```

process p_1 code:-

```
while (true)
  { Non_CS();
    while (turn != 1);
    [CS]
    turn = 0;
  }
```



Analysis :-

→ Assume turn = 0

→ Now P_0 is trying to enter critical section

→ Now turn != 0 is false, so P_0 enters into critical section.

→ Now P_0 be in critical section, if P_1 is toying to enter critical section then turn != 1 is true, so P_1 is not able to enter critical section.

→ When P_0 completes critical section, & makes turn = 1.

→ turn = 1 makes turn != 1 false & P_1 is able to enter critical section & P_0 is not able to enter critical section, So, mutual exclusion is satisfied.

→ Now, P_1 is leaving the critical section & makes turn = 0 & now after sometime P_1 is trying to enter critical section (critical section is free).

→ turn = 0 & hence turn != 1 is true & P_1 is not able to enter the critical section but P_0 is not in the critical section & is hindering P_1 to enter the critical section, hence progress is not satisfied.

Cosmos

A 2nd Defⁿ of Progress :-
Which process will go next into critical section is decided by those processes that want to go into critical section.
In this decision, the process in remainder section the process which is not interested to go into critical section should not take participation.

Date

12.06.12

Cosmos

(Similar to Strict Alteration)

Q.

Method used by P_1	Method used by P_2
while ($S_1 == S_2$);	while ($S_1 != S_2$);
[CS]	[CS]
$S_1 = S_2$;	$S_2 = (not)\ S_1$;

S_1 & S_2 are two shared (boolean) variables randomly assigned. Consider the methods used by process P_1 & P_2, Which of the below properties are satisfied:-

(a) Mutual Exclusion but not progress.
(b) Progress but not ~~mutual~~ mutual exclusion.
(c) both M.E. & progress.
(d) neither M.E. nor progress.

★ S_1 & S_2 can only have values 0 & 1.

Peterson's Solution:-

```
#define N 2
#define TRUE 1
#define FALSE 0
int turn;
int interested[N];

void enter_region(int process)
{ 1. int other;
  2. other = 1 - process;
  3. interested[process] = true;
  4. turn = process;
  5. while (turn == process && interested[other] == true);
}
```

→ the turn is of the 'process' & not of 'other' process but 'other' process is interested to enter

Q. Why turn == process?
Ans. Suppose the circumstances are initial, so interested[0] & [1]=FALSE & P0 enters into entry region & set interested[0]=TRUE & turn=0. Let suppose P_1 preempts P_0 just before statement 5, ∴ it sets interested[1]=TRUE & turn=1 & executes 5th inst^n, so turn==1 is true & interested[0]==true is also true, P_1 wont be able to enter CS, now for P_0, turn==0 is false & hence P_0 enters CS & when it leaves CS & execute interested[0]=FALSE, only then P_1 enters into CS, now let suppose P_1 enters into CS & P_0 is not interested, so interested[0]=FALSE, so P_1 enters into entry region again & set turn=1 & interested[1]=TRUE & executes 5th inst^n, now turn==1 is true but interested[0]==TRUE is false, so P_1 again enters into CS & hence satisfies progress.

CS

```
void Leave_region (int process)
{interested[process] = FALSE;
}
```

★ solution works only for 2 processes.
★ the initial values of interested[0] & interested[1] equals to FALSE.

Analysis:-

① First P_0 will be in critical section (assumption)
∴ process = 0
∴ other = 1
② In 3rd step interested[process] = interested[0] = true
③ In 4th step turn = process = 0.
④ In 5th step
while (turn == process && interested[other] == true)
(turn == process: true; interested[other] == true: false)
⑤ So, P_0 enters into critical section.
⑥ Now, let P_0 enter into cs be in critical section, we will check whether P_1 is allowed.
⑦ Now for P_1, we call enter_region.
⑧ For P_1, process = 1 & other = 0.
⑨ In 3rd step, interested[1] = true.
⑩ In 4th step, turn = 1.
⑪ In 5th step
while (turn == process && interested[other] == true)
(turn == process: true; interested[other] == true: true)
⑫ So, P_1 wont be able to enter critical section.
So, Mutual Exclusion is satisfied [when P_1 1st enters critical section & then P_0 trying to enter. So M.E. is satisfied only after checking both cases.]

Checking for progress

(13) Initial values for interested[0] & interested[1] equals false.

(14) For P0, Progress is satisfied.

(15) If interested[process] = FALSE is not executed, & process other process preempts, we assume that "process" is still in the critical section, & the "other" process can't enter the critical section. So progress is satisfied.

** If the no. of processes are countable or bounded, then bounded waiting is satisfied.

Hardware type Of Solutions

→ TSL instruction set :- [This is similar to Lock variable but here we load & change the value of flag in same instn.]
↳ test & set Lock

TSL Register flag :- Copies the current value of flag into the register & stores value of '1' into the flag in the single atomic cycle without any pre-emption.

* The soln. can be used for any no. of processes.

Ri :- seperate for diff processes

Entry Section
1. TSL Ri, M[flag]
2. CMP Ri, #0
3. JNZ to Step (1)
4. CS
4. Store M[flag], #0

This step takes place in 1 machine cycle.

These 3 instn. are atomic, which means no process can preempt the execution in btwn.

flag: 0 → CS is free; 1 → CS is busy.

Analysis :-
(1) Assuming CS is free & flag is 0.
(2) Assuming 2 processes in ready queue. (P1 & P2)
(3) Let P1 trying to execute.
(4) P1 executes 1st instn.
R1 ← 0
& flag is changed to 1 in

R1 ← 0
R2 ← 1
flag ← 1
R1 ←

⑤ Now, P_1 executes 2nd inst^n, so $R_1 = 0$, ∴ ZF = 1, so P_1 goes to critical section, now we will Check whether P_2 comes onto CS while P_1 is still in the CS.

⑥ P_2 executes 1st inst^n.
$R_2 \leftarrow 1$ & flag becomes 1

⑦ P_2 executes 2nd inst^n.
ZF = 0, ∴ When it executes 3rd inst^n, it will go back to 1st inst^n [Now, when we check for other conditions], & we always get only one process in CS, so Mutual Exclusion is satisfied.

⑧ Checking for progress:-
Let flag = 0, let execute P_1 1st

⑨ P_1 executing 1st inst^n
∴ $R_1 \leftarrow 0$, & flag = 1.

⑩ Assume P_1 is preempted after 1st inst^n by P_2.
Now, P_2 executes 1st inst^n
$R_2 \leftarrow 1$ & flag = 1

⑪ Now, P_2 executes 2nd inst^n
but for that CMP R_2, #0, ∴ ZF ≠ 1 & ZF = 0

⑫ P_2 executes 3rd inst^n & P_2 jumps back to inst^n 1, so P_2 is stopped by P_1 to enter the CS, though CS is free, but P_1 is interested in entering the CS & hence stops P_2 to enter CS.

⑬ Now, suppose P_2 preempts P_1 before execution of 4th step (as CS is free) but we assume that P_1 will remain in CS till 4th inst^n gets executed.
So, progress is satisfied.

⑭ Bounded waiting is not satisfied, because no. of processes are not bounded.

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

(a) The process which executes statement 2 1st.
(b) " " " " " 3 1st.
(c) " " " " " 4 1st
(d) We can't say anything, it all depends on situation.

Reason:- Initially
① interested[0] = interested[1] = false
② Now, executing P_0 first.
∴ other = 1 & process = 0!
In 3rd step, interested[0] = true.
& assume P_1 executes 3rd step after it, ∴ interested[1] = true
③ Now, for 4th step
let P_1 execute it 1st
∴ turn = 1
& now, P_0 execute it
∴ turn = 0.
④ Now, both checking 5th inst^n
Let P_1 execute it 1st
∴ while (turn == 1 && interested[other] == true);
false as turn = 0

Test & Set

classmate

- Deadlock can't occur.
- But Starvation can occur when there are ~~let suppose~~ 10 processes, then it may happen that process P_1 will Test & set the flag always & hence is always executed, & hence can cause starvation.

So, P_1 is executed 1st because it executes 4th inst^n

Using Preemption

→ For P_0 process

- When it executes 3rd inst^n
  interested[0] = true.
- Now, P_1 preempts it.
- So, P_1 executes now in 3rd inst^n
  interested[1] = true.

Now, P_1 executes 4th inst^n
∴ turn = 1

Now, P_1 executes 5th inst^n
but P_1 is stopped at 5th inst^n.

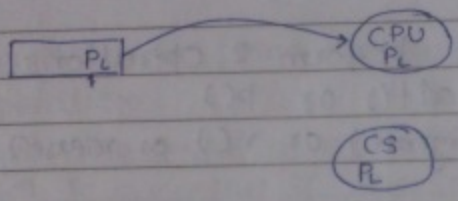
- Now, P_0 comes back after 3rd inst^n,
  ∴ P_0 executes 4th inst^n
  ∴ turn = 0
- Now, P_0 executes 5th inst^n & gets struck there,
- & when P_1 executes 5th inst^n now, & will get its CS executed.

Soln.	M.E.	Progress	Bounded Waiting	
1. Lock variable	X	✓	X	
2. Strict alternate	✓	X	✓	
3. Peterson (without applicable for any 2 processes)	✓	✓	✓	(best)
4. TSL	✓	✓	X	

Cosmos

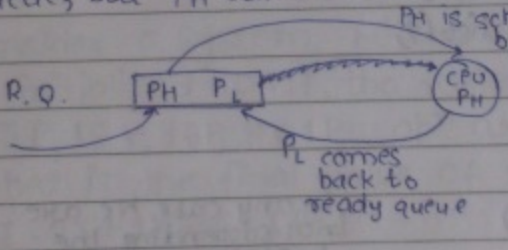
Priority Inversion Problem

1. Assume that we have only one process P_L with low priority in ready queue.



since, only one process, so it will be scheduled for CPU & P_L will be in the critical section.

2. Now, suppose a high priority process preempts P_L when P_L is executing its CS.
3. So, P_H is scheduled to CPU & P_L will come in ready queue, but P_H can't enter CS because P_L has locked it.



Operating System Soln.

Semaphore :-

- It is an integer variable which is used by various processes in a mutual exclusive manner to achieve synchronisation.
- Improper usage of semaphore will also give improper results.

Cosmos

Semaphores

- Counting Semaphores [takes value from $-\infty$ to $+\infty$]
- Binary Semaphores [takes value 0 or 1]

The semaphores perform 2 operations:-

1. Down() or Wait() or P()
2. Up() or Signal() or V() or release()

Counting Semaphore:-

```
1. Down(Semaphore S)
   { S.value = S.value - 1;
     if (S.value < 0)
       { Block the process & place its PCB in the suspend_list();
       }
   }

Up (Semaphore S)                    In any case, we are not suspending the process.
   { S.value = S.value + 1;
     if (S.value <= 0)
       { select a process from suspend list() & wakeup();
       }
   }
```

wakeup(): the process is allowed to perform the remaining code of down op.

★ After performing Down operation, if process is getting suspended, then it is called as unsuccesful down operation.
If down operation is unsuccesful, the process will not continue the execution. [i.e. the remai in down() afte

★ After performing down operation, if process

is not getting suspended, then it is called as succesful down operation, if it is succesful down opⁿ, it will continue further execution. [i.e. the code after the if block in the down() opⁿ.]

★ So, the process will come in suspend-list in unsuccesful down operation.

★ There is no unsuccesful up operation, it is always succesful

★ Down opⁿ is succesful if the semaphore value is greater than equal to 1 (& not 0).

★ If Semaphore value is +7, this means we can perform 7 succesful down opⁿ.

★ If Semaphore value is -7, this means we have 7 suspended processes.

Q. Consider a system where counting semaphore value is initialized to +17, the various semaphore opⁿ like 23P, 18V, 16P, 14V, 1P opⁿ are performed, then what is the final value of semaphore

Ans. +17 - 23 + 18 - 16 + 14 - 1
= +9

the value of semaphore is 9.

Binary Semaphore

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

→ if semaphore value is 1, then we will continue the code after the else block.

but if it is 0, the process will be put into suspend list

Cosmos

```
Up(Semaphore S)
{ if (suspended list() is empty) → if suspended list is empty, then there are no suspended processes & hence we can't wake any process.
    { s.value = 1;
  else
    { select a process from suspended list & wakeup(); } if it is not empty, then we will wake any process & continue the execution.
    }
}
```

- The down opⁿ is succesful only when semaphore value is 1.
- There is no unsuccesful up operation.

Q. Each process i = 1 to 9, executes the below code :-

```
Repeat
    p(mutex);
    CS
    V(mutex);
forever
```

The process p_{10} executes below code

```
Repeat
    v(mutex);
    CS
    v(mutex);
forever
```

The initial value of binary semaphore mutex = 1 what is the max. no. of process that may be present inside the critical section at any pt. of time?

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

Cosmos

$P_1 P_{10} P_2$ ...
$P_1 P_2 P_3 P_4 P_{10} P_5$

My Answer :- (b) 3

Reason :- Suppose 1st process enters

Correct answer :- (d) 10

- Initially P_1 is in CS, & all the other processes P_2 to P_9 are in suspend list.
- When code for P_{10} executes then it executes v(), ∴ $P_1 P_2$ is in CS & P_{10} also comes in CS ∴ $P_1 P_2 P_{10}$ is in CS.

Now, when P_{10} comes out of CS, & calls v()
∴ $P_1 P_2 P_3$ is in CS.
& now code repeats again, & v() is called again
∴ $P_1 P_2 P_3 P_4$ is in CS
& P_{10} also comes into CS
∴ $P_1 P_2 P_3 P_4 P_{10}$ are in CS.
∴ similarly all the processes can come into CS.

Q. Now if change P_{10} code as :-

```
Repeat
    v(muter);
    [CS]
    p(muter);
forever
```

→ Now, max. no. of processes in CS are 3.

Q Now, if we change P_{10} code as :-

```
Repeat
    p(muter);
    [CS]
forever v(muter);
```

then only 1 process can be in CS at any pt of time.

Q. Consider two concurrent processes 'P' & 'Q' executes their respective codes

P :-	Q :-
while (true)	while (true)
{ w: ____	{ y: ____
print('0');	print('1');
print('0');	print('1');
x: ____	z: ____
}	}

What should be the binary semaphore opn on w, x, y, z & what should be the initial values of binary semaphore variables 'S' & 'T' to get o/p as 0011001100...

(a) w = p(T), x = v(T), y = p(S), z = v(S), S = T = 1
(b) w = p(T), x = v(T), y = p(S), z = v(S), S = 1, T = 0
(c) w = p(T), x = v(S), y = p(S), z = v(T), S = T = 1
(d) w = p(T), x = v(S), y = p(S), z = v(T), T = 1, S = 0

S=1
00

★ if we have S=1, then 11 can start the o/p as well.
∴ T=1 & S=0.

Q. Which of the below ensures that o/p string never contain a substring of 01^n0 (or) 10^n1 where n is odd.?

(a) w = p(S), x = v(S), y = p(T), z = v(T), S = T = 1
(b) w = p(S), x = v(T), y = p(T), z = v(S), S = T = 1
(c) w = p(S), x = v(S), y = p(S), z = v(S), S = 1
(d) w = v(S), x = v(T), y = p(S), z = p(T), S = T = 1

Ans. (a), (b), (d) are printing 010, (c) is not printing 010.

Q. Page 45, Q-20.

Case 1:- Starting with P_0

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

∴ O/P :- 000

Case 2:- $P_0 \rightarrow P_1, P_2 \rightarrow P_0$ (when both P_1 & P_2 are executed parallely)

00

$S_0 = \cancel{X}\cancel{0}\cancel{X}\cancel{X}\cancel{0}0$

$S_1 = \cancel{0}\cancel{X}\cancel{0}1$

$S_2 = \cancel{0}\cancel{X}\cancel{0}1$

Classical problems

Producer-Consumer:-

```
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(emptyk/sfull);
    }
}
```

* Mutex is a binary semaphore variable used by the producer & consumer to access the buffer in a mutual exclusive manner.
* Empty is a counting semaphore variable representing the no. of empty slots in the buffer at any point of time.
* Full:- a counting semaphore variable representing no. of filled slots in the buffer at any point of time.

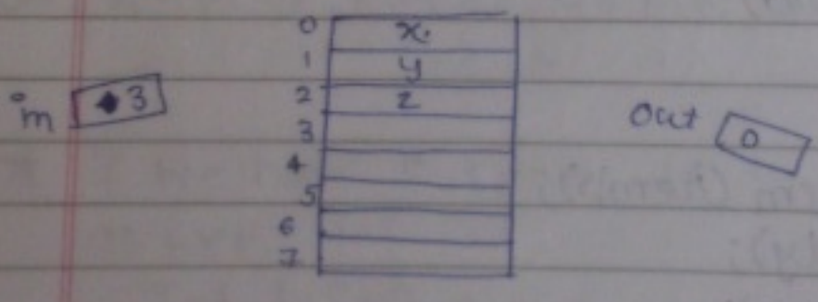
```
void consumer (void)
{ int ItemC;
  while(true)
  { down (full);
    down (mutex);
    ItemC = Buffer[out];
    out = (out+1) mod N;
    up (mutex);
    up(empty);
    process_item (itemc);
  }
}
```


Analysis

N=8

producer

consumer



In this example:-

→ empty = 5

↓ full = 3

→ We are producing an item

down(empty)

∴ empty = 4

↓ down(mutex)

∴ mutex = 0.

Cosmos

→ Now,
in = (in+1) mod N
∴ in = 4.

→ Now, up (mutex)
∴ mutex = 1
& up (full)
∴ full++;
∴ full = 4.

→ Now, we execute consumer program
down(full)
∴ full = 3
& down (mutex)
∴ mutex = 0

→ Now;
out = (out+1) Mod N
∴ out = 1

→ Now up(mutex) ∴ mutex = 0

→ & now up(empty), ∴ empty = 4

★ If Buffer is full
Empty = 0
Full = 8
so, producer is not allowed to produce by statement down(empty)
∴ the producer will halt here.

★ If Buffer is empty
Empty = 8
Full = 0
so consumer is not allowed to consume by down(full).
∴ consumer will halt.

Q. If we interchange down(empty) & down (mutex), then

→ We encounter down(mutex) 1st which will be executed, & mutex = 0, & goes to next inst^{n}.

→ down(empty) if empty = 0, then producer halts here.

→ The consumer will now executes down(full) which makes the exec. of next inst^{n} which is down(mutex) but mutex = 0, so consumer also halts,

→ So, both producer & consumer halts & hence deadlock.

Readers-Writers Problem :-

```
int rc = 0;        → readers count : it represents the no. of readers present in the d/bat (database) at any point of time.
semaphore mutex = 1; → it is a binary semaphore used by the readers.
semaphore db = 1;  → It is also a binary semaphores used by readers & writers.
void Reader(void)
{ while(TRUE)
    { down(mutex);
      rc = rc + 1;
      if (rc == 1) down(db);
      up(mutex);
      [DB]
      down(mutex);
      rc = rc - 1;
      if (rc == 0) up(db);
      up(mutex);
    }
}
```

```
void writer(void)
{ while(TRUE)
    {
      down(db);
        DB
      up(db);
    }
}
```


If reader is in the database, then writer is not allowed to update the database.

* If reader is in the database, then readers are allowed.
* If writer is in the d/b, then readers are not allowed
* " " " " " " , " writers " " ".

Analysis

rc = 0, mutex = 1, db = 1

① Assume 1st reader is entering into d/b.
② down(mutex)
∴ mutex = 0
③ Now, rc = rc + 1 = 1
④ Now, checking rc == 1, which is true,
∴ down(db), ∴ db = 1
⑤ Now, up(mutex), ∴ mutex = 1
⑥ Now, 1st reader is in the db.
⑦ Now, we will check whether writer can come in the d/b or not.
⑧ So, we will execute writer code.
⑨ So, writer has down(db)
but db = 0, hence writer is suspended.

(10) Now, we will check whether 2nd reader can enter the d/b.

(11) So, we execute reader's code.
down (mutex)
∴ mutex = 0,
& rc = rc + 1 = 2
& (rc == 1) is false

(12) up(mutex)
mutex = 1
& hence the 2nd reader enters the d/b which is acc. to the problem.

---

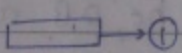
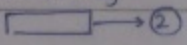
(1) Now, 1st writer enter the d/b, so writer's code executes.

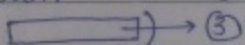
(2) ∴ down(db)
∴ db = 0
& writer is in the d/b.

(3) Now, we will check while writer is in db, is the reader allowed?, so we execute reader's code.

(4) down (mutex)
∴ mutex = 0
& rc = rc + 1 = 1

(5) & (rc == 1) is true
& down (db)
[since db = 0 prior to down(db), ∴ the reader will be suspended.]

Q. int R=0, W=0;
Semaphore mutex=1;
void Reader(void)
{ L1: down(mutex);
if(W==1)
{  up(mutex)
goto L1;
}
else
{ R=R+1;
 up(mutex)
}
DB
down(mutex);
R=R-1;
up(mutex);
}

void writer(void)
{ L2: down(mutex);
if( R≥1 or W==1
{ up(mutex);
goto L2;
}
W=1;
up(mutex);
DB
down(mutex);
W=0;
up(mutex);
}

Q. What should be the values of blanks 1,2,3 to synchronise classical readers & writers?
(a) down(mutex), up(mutex), W==1
(b) up(mutex), up(mutex), W==1
(c) down(mutex), up(mutex), R≥1 or W==1
(d) up(mutex), up(mutex), R≥1 or W==1

Cosmos

Q. What happens if we interchange w=1 & up(mutex) in writer's code.

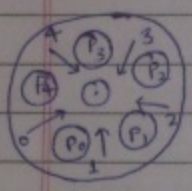
(a) Soln works fine, no problem at all.

(b) Only multiple writers are allowed.

(c) readers & writers both can be in DB at same time.

(d) None of them.

Dining Philosopher




```
void philosopher(int i)        → philosopher's number
{ while(true)
    { thinking();
      take_fork(i);            → pick the left fork
      take_fork((i+1)%N);      → pick the right fork
      eat();
      put_fork(i);             → put the left fork back on table.
      put_fork((i+1)%N);       → put the right fork back on table.
    }
}
```

P_0 waiting for P_1 to put the right fork

P_1 " " P_2 " " " "

P_2 P_3 " " P_4 " " " "

P_3 P_4 " " P_0 " " " "

P_4

So, it's a deadlock.

Cosmos

```
#define N 5
#define LEFT (i+N)%N  → left philosopher
#define RIGHT ((i+1)%N) → right philosopher
#define Thinking 0
#define HUNGRY 1
#define EATING 2
Semaphore mutex=1;  → mutex is binary semaphore variable used by the philosophers in a mutual exclusive manner.
Semaphore S[N]; // all s[i]s are initialised to 0   (array of semaphores)
int state[N]; // array to keep track of everyone's state.
void philosopher(int i)
 {while(true)
    {thinking();
     take_forks(i);
     eat();
     put_forks(i);
    }
 }
take_forks(int i)
 {down(mutex);
  state[i]=HUNGRY;
  test(i);
  up(mutex);
  down(s[i]);
 }
put_forks(int i)
 {down(mutex);
  state[i]=THINKING;
  test(LEFT);  → checking whether left philosopher can go into eating state.
  test(RIGHT);  → checking whether right philosopher into eating state.
  up(mutex);
 }
```

test

Q If we interchange up(mutex) & down(s[i] then down(s[i]) will suspend the philosopher who it is hungry & can't eat, & hence up(mutex) won't be executed, & those philosophers that are eating can't put the fork back because put_forks function's 1st statement is down(mutex), & the philosopher is suspended & will remain in eating state.

Cosmos

```
void test (int i)
{ if (state[i]== HUNGRY && state[LEFT]!= EATNG &&
   state[RIGHT]!=EATNG)
  { state[i]= EATING;
    up(s[i]);
  }
}
```

Q. Let P[0]... P[4] be processes & M[0]... M[4] be binary semaphores, mutex variable is initialised to 1.

Each process P_i executes the below code:-

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

Consider the statements below:-

I. M.E. is satisfied

II. M.E. is not satisfied

III. It is possible for deadlock

Which of the above statements are true

(a) only I

(b) only I & III

(c) only II & III

(d) only II

★ Deadlock:- When all the processes tries to enter CS, then 1st wait statement is succesful, but 2nd wait opn causes all processes to be in suspended state.

Date

24.06.12

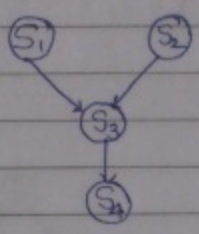
Cosmos

Concurrent Programming

- It is possible when we have multi-processors system.

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

Dependency Graph



S_1 & S_2 can be run concurrently (i.e. independently) or parallely.

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

S_i & S_j can be run concurrently or parallely :-

1. $R(S_i) \cap W(S_j) = \phi$
2. $W(S_i) \cap R(S_j) = \phi$
3. $W(S_i) \cap W(S_j) = \phi$

R : - read set
W : - write set

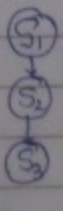
To write programs parallely:

parallel → par begin (or) co-begin ← concurrent

par end (or) co-end

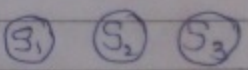
If we write:-

begin
S_1:
S_2:
S_3:
end



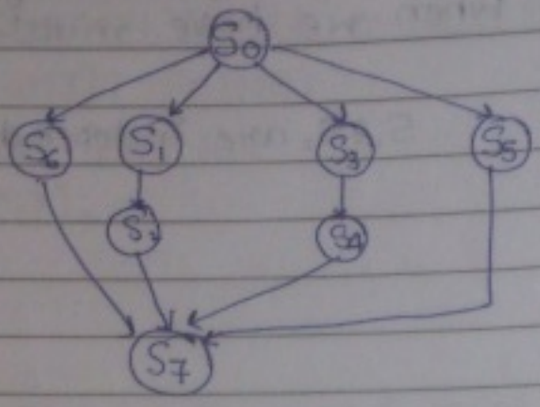
If we write:-

par begin
S_1:
S_2:
S_3:
par end

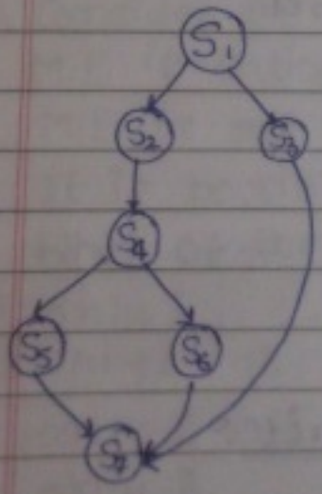


Cosmos

example:- S_0;
par begin
begin
S_1;
S_2;
end
begin
S_3;
S_4;
end
S_5;
S_6;
par end
S_7;



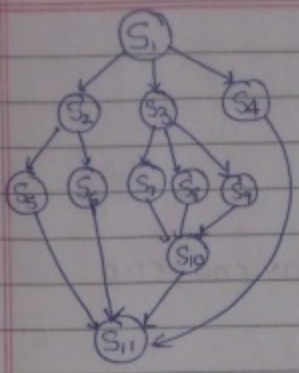
Q. Dependency Graph



begin
S_1;
par begin
begin
S_2;
S_4;
par begin
S_5;
S_6;
par end
end
S_3;
par end
S_7;
end

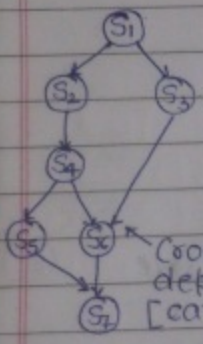
Cosmos

Q.




```
begin
S1;
par begin
  begin
    S2;
    par begin
      S5;
      S6;
    par end
  end
  begin
    S3;
    par begin
      S7;
      S8;
      S9;
    par end
    S10;
  end
  S4;
par end;
S11;
end
```

Q.




```
begin
S1;
par begin
S3;
```

★ It is not possible to write concurrent programs for all dependency graphs using par-begin & par-end.
It is very much possible with help of semaphore op^n.

★ all binary semaphores a,b,c,d,e,f,g,h are initialised to 0.

par begin

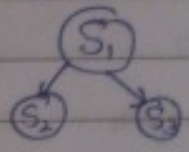
← begin S_1: par begin v(a), v(b), par end, end
begin p(a), S_2, v(c), end
begin p(b), S_3, v(d), end
begin p(c), S_4, par begin v(e), v(f), par end, end
begin p(e), S_5, v(g), end

Cosmos

begin p(d), p(f), S_6, v(h), end
begin p(g), p(h), S_2, end
par end

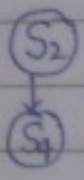
→ In 1st inst^n

- begin S_1 : par begin v(a), v(b), par end, end



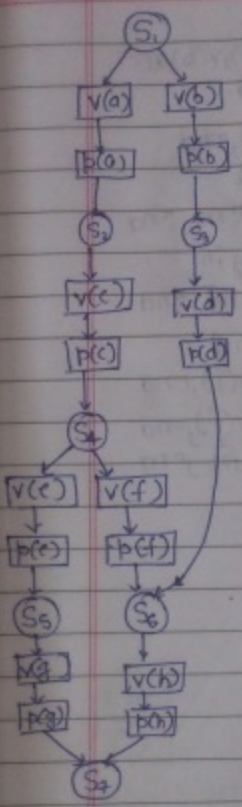
S_2 & S_3 can only be executed when S_1 is completed.
& S_2 & S_3 are in parallel,
∴ we perform v(a) & v(b)
for S_2 for S_3

- begin p(a), S_2, v(c), end
for S_4



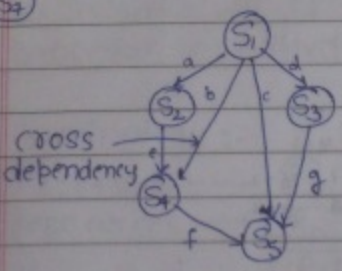
p(a) → means only after completion of S_1, we can execute S_2.
v(c) → we are upping semaphore c, which means S_4 can only be executed when S_2 is executed.

- & so on...



Cosmos

Q.



par begin

begin S1: par begin V(a), V(b), V(e), V(d), par end, end

begin : P(a), S2, V(e), end

~~begin : P(b), S~~

begin : P(e), P(b), S4, V(f), end

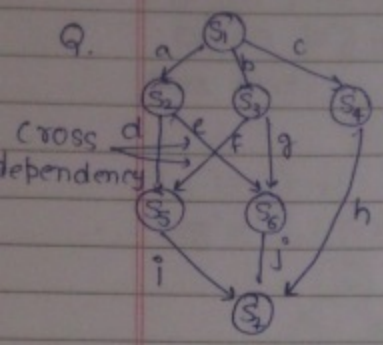
begin : P(d), S3, V(g), end

begin : P(c), P(g), P(f), S5, end

par end

Cosmos

Q.



par begin
begin S_1: par begin V(a), V(b), V(c), par end, end
~~begin : P(a), S_2, V(d), end~~
begin : P(a), S_2, par begin V(d), V(e), par end, end
begin : P(b), S_3, par begin V(f), V(g), par end, end
begin : P(c), S_4, V(h), end
begin : P(d), P(f), S_5, V(i), end
begin : P(g), P(e), S_6, V(j), end
begin : P(i), P(j), P(h), S_7, end
par end

Q. int x=0, y=0;

par begin
- begin
  - x=1;
  - y=y+x;
  - end
- begin
  - y=2;
  - x=x+3;
  - end

par end

x=1 | y=2
y=1 | x=5

x=1
y

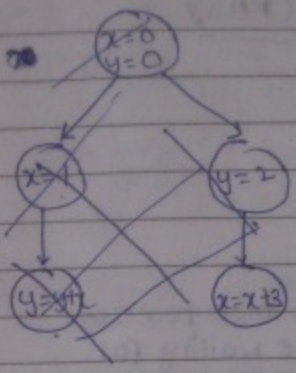
x=1 y=2
y=3
x=4

x=1
y=1
x=1 y=1
x=4 y=4

What are the possible values of x & y after completion of program

I x=1, y=2
II x=1, y=3 ✓
III x=4, y=6 ✓

Ans. II & III



II → y = 2 | x: 1 | y: 3
x = x+3
x = 1
y = y+x

II → x = 1 | x: 4 | y: 6
y = 2
x = x+3
y = y+x

Deadlock

Definition:-
If two or more processes are waiting on some event to happen which never happen, then we say those processes are involved in deadlock.

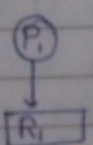
Terminology :-

① Processes → P_1 P_2

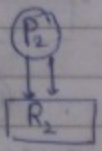
② Resources → R_1 R_2

R_1 : → means 2 instances of R_1

③ Requesting

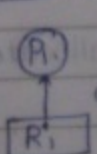


process P_1 is requesting for 1 instance of R_1

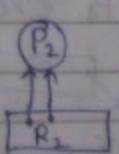


process P_2 is requesting for 2 instances of R_2

④ Allocation



P_1 allocated one instance of R_1

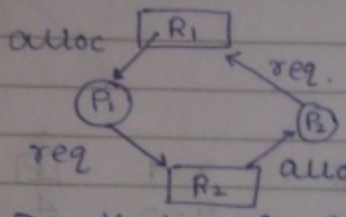


P_2 allocated 2 instances of R_2.

Resource Allocation Graph (RAG)

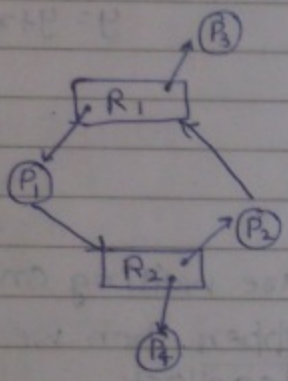
RAG = G(V, E)
- V: vertices → processes & resources
- E: edges → request & allocation edge

ample:-



1. Two or more processes
2. They are waiting for some event to happen
3. Event never happens

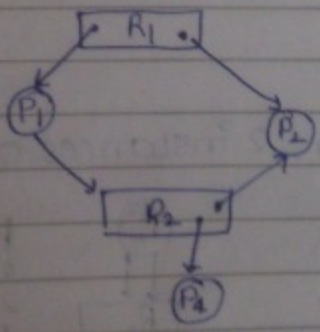
Deadlock exist in RAG.



1. Two or more processes
2. P_1 & P_2 are waiting for R_2 & R_1 respec.
3. But R_1 & R_2 can be allocated to P_2 & P_1 respec. at some pt. of time.

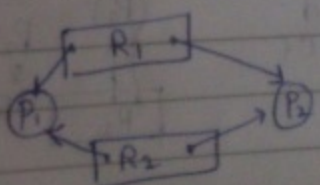
So no deadlock.

When P_3 executes



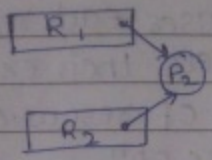
R_1 is given to P_2

When P_4 executes

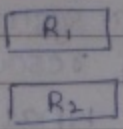


R_2 is allocated to P_1.

When P_1 executes



When P_2 executes



Life-Cycle of Resource Request & Release :-

1. The process request for the resource.
2. O.S. validates the request of the process. (if the resource is not available for in the system, i.e. if there is no printer present & a process is requesting printer, then it is an invalid request & is rejected by OS.)
3. O.S. checks the availability of the resource.
4. If the resource is available it will be allocated to the process, otherwise process has to wait.
5. The process will go into execution only if all the Resources required by the process at that pt. of time are allocated.
6. The process release all the resources after completion of the process execution.

→ Dread

Q. Consider a system which has n processes & 6 tape drives. If each process requires 2 tape drives, complete their execution, then what is the max. value of n which ensures deadlock free operation

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

Cosmos

Q. Consider a system with 3 processes each require 2 units of resource R to complete their execution. Then what is the min. no. of resources require to ensure deadlock free opⁿ.

P_1 P_2 P_3

R R R

R

Ans. 4 resources.

2 can't be the right answer, because

P_1 P_2 P_3 → deadlock

R R

3 can't be the right answer, because

P_1 P_2 P_3

R R R → deadlock

Q. Consider a system with n processes & 6 tape drives. If each process requires 3 tape drives to complete their execution, then what is the max. value of n for deadlock free opⁿ.

R R R R R R

P_1 P_2 P_3 P_4 P_1 P_2

R R R R R R

R R R R

R R

Ans. if we have max. 4 processes:

P_1 P_2 P_3 P_4 → deadlock

R R R R

R R

if we have 3 processes:-

P_1 P_2 P_3
R R R → deadlock
R R R

if we have 2 processes:-

P_1 P_2
R R → no deadlock
R R
R R

∴ ans. is (2).

Q. Consider a system with 3 processes P_1, P_2, P_3, the peak demand of each process is 5, 9, 13 respec. for resource R. Then what is the min. no. of resources req. for deadlock free opn.

(a) 22
(b) 23
(c) 24
(d) 25

P_1 P_2 P_3
0 13
15

$P_1 \rightarrow 5$
$P_2 \rightarrow 9$
$P_3 \rightarrow 13$

if we give 4 to P_1 & 8 to P_2 & 12 to P_3, then
4+8+12=24
it can cause deadlock
so, if we give 1 additional resource, then there will be no deadlock
∴ 24+1=(25)

Cosmos

Page 48 Q11

P_1	P_2	P_3	P_4
4	3	7	8

→ 20 identical resources

$P_1 \rightarrow 4$
$P_2 \rightarrow 3$
$P_3 \rightarrow 7$
$P_4 \rightarrow ?$

total is 20.

$3+2+6 = 11$
$20-11 = 9$

$P_1 \rightarrow 4 \longrightarrow 3$
$P_2 \rightarrow 3 \longrightarrow 2$
$P_3 \rightarrow 7 \longrightarrow 6$
$P_4 \rightarrow 9 \longrightarrow 8$

$19 + 1$ ← 1 additional resource for deadlock free opn

Page 49 Q18

S_i

$P_1 \; P_2 \; P_3 \; \ldots \; P_n$

$P_1 \; P_2 \; P_3 \; P_4$

Sufficient condition :-
least upper bound which satisfy the condition.

$P_1 \; P_2 \; P_3 \; P_4 \; \ldots \; P_n$

$\forall_i S_i < m$
$m = 10$
$n = 100$

$P_1 \rightarrow 9$
$P_2 \rightarrow 9$
$P_3 \rightarrow 9$
.
.
.
$P_{100} \rightarrow 9$

$\forall_i S_i < n$
↓
no meaning at all

but $m = 10$ $\quad 900$
↳ deadlock

Cosmos

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

P_1 P_2 P_3 P_4 ... P_n

m m m $m*n$

$(S_1 + S_2 \dots S_n) < n+m$

P_1 P_2 P_3 P_4 P_n

(S_1-1) (S_2-1) (S_3-1) (S_4-1) (S_n-1) → for deadlock

$(S_1-1) + (S_2-1) + \dots + (S_n-1) <$

Q15 Page 19

$n \rightarrow$ process

$i \rightarrow$ holding x_i of R

$(n-2) \rightarrow$ are competing

$y_p = y_q = 0$

$x_0 + x_1 + \dots + x_n = R$

$\min(x_p, x_q) < \max y_k$

Deadlock Characteristics:

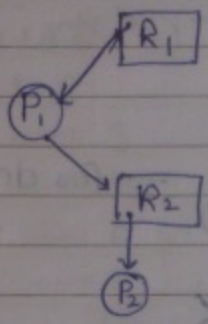
1. Mutual Exclusion
2. Hold & Wait
3. No pre-emption
4. Circular Wait

Mutual Exclusion

- The resource has to be allocated to only one process or it is freely available.
- There should be a 1-to-1 relationship b/w the resource & the process

Hold & Wait

- The process is holding some resources & waiting on some other resource simultaneously.

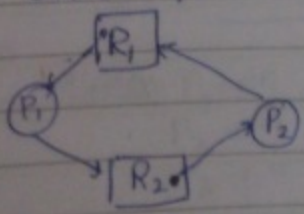


No pre-emption

- The resource has to be voluntarily released by the process after the completion of execution.
- It is not allowed to pre-empt the resource forcefully from the process.

Circular Wait

- The processes are circularly waiting on each other for the resources.



★ If only all the processes above 4 cond^{n} are occuring simultaneously in the system, then

definitely there exist a deadlock.

Deadlock Prevention

1. Mutual Exclusion:-

It is not possible to dissatisfy mutual exclusion always because of sharable & non-sharable resources.

→ File is sharable resource
it can be read by two or more processes.
→ but Printer is Non-sharable resource.

2 Hold & Wait :-

(i) Allocate all the resources required by the process before the start of execution.
If we follow this strategy we will have low device utilization.

(ii) The process should release all the existing resources before making a new request.
If we follow this strategy, then it is possible for starvation.
(The process is not under execution).

3 No pre-emption:-

Process P_1 requesting resource R_1

- if R_1 is available, then it will be allocated to P_1
- if R_1 is busy, & it is allocated to other process P_2, then
  - if P_2 is in execution, then P_1 will wait.
  - if P_2 is not in execution & it is waiting on for other resource R_2, then we will pre-empt P_2 & take R_1 from it & allo it to P_1.

4. Circular Wait

Every resource will be assigned with a numerical no. The process can request for the resource only in the increasing order of enumeration.

e.g.

$P_1 \rightarrow R_7$	$P_1 \rightarrow R_5$	$P_1 \rightarrow R_9$
granted	rejected	granted.

Deadlock Avoidance

Banker's Algorithm.

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

Total:-

A	B	C
↓	↓	↓
10	5	7

Remaining Need = Maximum Need - Current Allocation

Current Available = Total Available - Current Allocation

★ Now, we need to identify whether system is in safe state or unsafe state,
if it is unsafe state, then deadlock can occur.
(only possibility)
may or may not occur

★ We can multiple safe sequences.

① If we can satisfy the remaining need of all the processes with current available resources in som order then the system is said to be in safe state, otherwise System is in unsafe state.

② If System is in unsafe state, then deadlock may occur.

③ The order in which we satisfy the remaining need of all the processes is called as safe sequence.

④ The safe sequence can't be unique, we can have multible safe sequences.

⑤ The unsafe state purely depends on behaviour of the processes.

⑥ Deadlock avoidance is more restrictive than deadlo prevention.

	5	4	3
P_3	7	4	3
P_1	~~7~~	~~5~~	~~3~~
P_0	9	5	5
P_2			
P_4			

ns.→ P_1

	2	1	0
	1	2	2
	2	0	0
	5	3	2

now available resources =
current allocation + current available
= 2,0,0 + 3,3,2
= 5,3,2

Now, P_3
available resources = 5,3,2 + 2,1,1 = 7,4,3

Now P_0
available resources = 7,4,3 + 0,1,0 = 7,5,3

Now P_2
available resources = 7,5,3 + 3,0,2 = 10,5,5

Now P_4

∴ available resources = 10,5,5 + 0,0,2 = 10,5,7

∴ Safe Sequence → $P_1 P_3 P_0 P_2 P_4$

Page 47
Q3.

P: P_1, P_0, P_2, P_3

P_1 → available now 4,3,1 + 2,1,2 = 6,4,3

P_0 → available now

	Need A	B	C
P_0	6	2	0
P_1	1	3	0
P_2	1	0	2
P_3	2	0	4

P_1: 4,3,1 + 2,1,2 = 6,4,3

P_0: 6,4,3 + 0,3,4 = 6,7,7

P_2: 6,7,7 + 0,0,2 = 6,7,9

P_3: 4,3,1 + 2,1,2 = 6,3,5

Q: P_1, P_2, P_3, P_0

P_1 → available now 4,3,1 + 2,1,2 = 6,4,3

P_2 → available now 6,4,3 + 0,0,2 = 6,4,5

P_3 → available now 6,4,5

Q: P_1, P_2, P_3, P_0

R: 4,3,1 + 2,1,2 = 6,4,3 — P_1 P_3

P_1	P_2	P_3	P_4	P_0	P_5
1	1	1	1		✗
1	1	1	1		
✗	✗	✗			

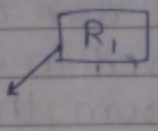
2
3
5
10

Cosmos

Deadlock Detection

2/3/5 10

1. If all the resources are of single instance type
→ If Cycle exists in RAG, then deadlock exists



- If all the resources are single instance, then cycle in RAG is necessary & sufficient condⁿ for occurence of deadlock.
- If all the resources aren't single instance, then cycle in RAG is necessary but not sufficient condⁿ for occurence of deadlock.
- If all the resources are of multiple instance type, then the Banker's algo is used. Whether we can satisfy the remaining need of all the processes with current available resources.
- If we can satisfy, we can say that the deadlock doesn't exist, otherwise it exist in the system.

Deadlock Recovery :-

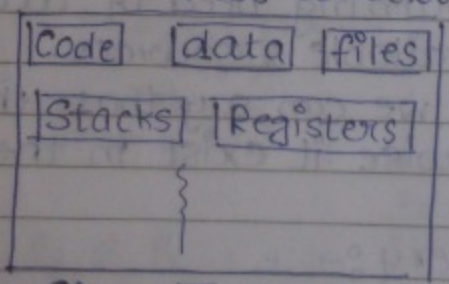
1. Killing the process
   - Kill all the processes involved in the deadlock.
   - Kill one after the other. Criteria:-
     (a) Priority (generally low priority process is selected first to be killed.)
     (b) % of process completion :-
     (the process which is more completed is not killed)
     (c) no. of resources the process is holding (kill those processes which is holding more no. of resources)

2. Resource Pre-emption:-
The resource will be pre-empted from the process involved in deadlock.
The pre-empted resources are allocated to other processes involved in deadlock to recover the system from the deadlock

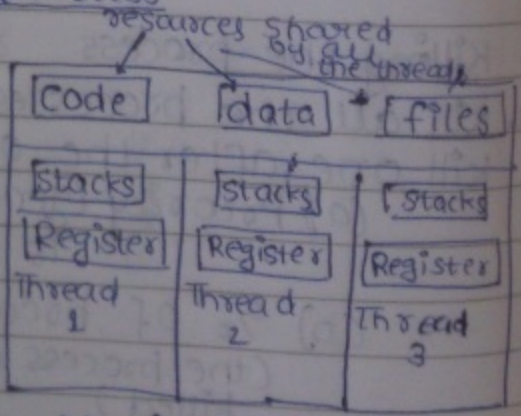
3. Ostrich Algorithm
→ Ignore the deadlock.

Threads

efn :- Light weight process.
↳ less context



Single Threaded Process



Multi-threaded process

Cosmos

Advantages Of Thread :-

1. Responsiveness :- The thread will improve the responsiveness If one thread has completed the execution, the o/p will be responded early as compared to the completion of the process.
2. Less Context Switching time :- The thread has less context, ∴ less context switching time.
   - The context switching time will be less in threads compared to the process.
3. Resource Sharing :- The resources of the process will be shared among all the threads within the process.
4. Effective utilization of multiprocessor systems :- If multiprocessor system is used then diff threads of a process can be scheduled onto diff. processors, so that the process execution will be faster.
5. Implementing the threads is very economical. There are various programming languages which supports implementing the thread. e.g. Java
6. Enhanced Throughput of the system :- If the thread is considered as a job, then the no. of jobs completed /unit time increase, hence throughput of the system is enhanced.

Threads → User-level Threads
Threads → Kernel-level Threads

- The OS can't recognize User-level threads

Cosmos

	User level Threads	Kernel level Threads
1.	The OS can't recognise user-level threads. The OS views user-level thread as a process only. (if one user level thread wants to perform I/O op^n, then the entire process is moved into wait state)	1. These threads are created by the OS & recognised by the OS. (if one kernel level thread is performing blocking system call, then another thread will continue the execution)
2.	These threads are dependent threads.	2. Kernel level are ~~defined~~ designed as independent threads.
3.	Less Context	3. More Context.
4.	Doesn't require h/w support.	4. They require h/w support.

William Stallings

Memory Management

Main Memory / Physical Mem / Primary Mem / RAM

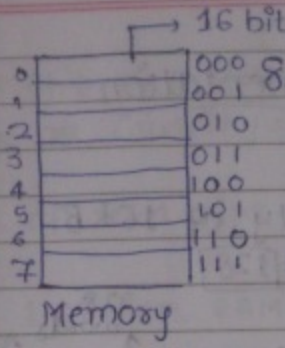
Functionality:- Allocation & Deallocation of memory to the processes.

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

$2^{10} \approx 10^3 \rightarrow 1K$
$2^{20} \approx 10^6 \rightarrow 1M$
$2^{30} \approx 10^9 \rightarrow 1G$
$2^{40} \approx 10^{12} \rightarrow 1T$

Memory → Byte Addressable
Memory → Word Addressable.

Cosmos



capacity of memory = 8 × 16 bits
= $\frac{8 \times 16}{8}$ bytes
= 16 bytes

Capacity of memory = no. of words in memory × word size

Q. Consider a system which has 128 K words, where each word is having size of 64 bits, then what is the capacity of memory in Bytes?

Ans. Capacity = 128K × 8 Bytes
= 1024 KBytes [$2^{10} \times 2^{10}$ Bytes]
= 1 M Bytes [2^{20} Bytes]

$2^9 \times 2^{10} \times 2^4$

Q. Consider a system having 512 M words, each word is having size of 16 bytes

Ans. $2^9 \times 2^{20} \times 2^4$ Bytes
= $2^{30} \times 2^3$ Bytes → 1 G Bytes
= 8 G Bytes [8×2^{30} Bytes]

Q. Consider a system where 25 bits are used to represent the words of the mem., each word has the size of 32 bits

Ans. = $2^{25} \times 4$ Bytes
= 2^{27} Bytes
= $2^7 \times 2^{20}$ Bytes
= 128 M Bytes

Cosmos

RAM chip implementation

RAM chip size = 128 Bytes
organize the mem. capacity of 16 KB

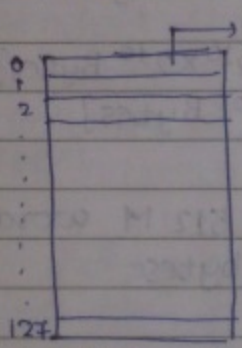
$$\frac{2^4 \times 2^{10}}{2^7 2^3} = 2^7 = \boxed{128}$$

(a) How many RAM chips are req.

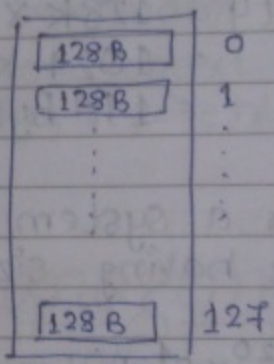
$$\frac{2^4 \times 2^{10}\ \cancel{B}}{2^7\ \cancel{B}} = \boxed{128}\ \text{RAM chips.}$$

128 words ↑ each word of 1 Byte
128 B RAM chip size

(b) Draw Mem. org. map.



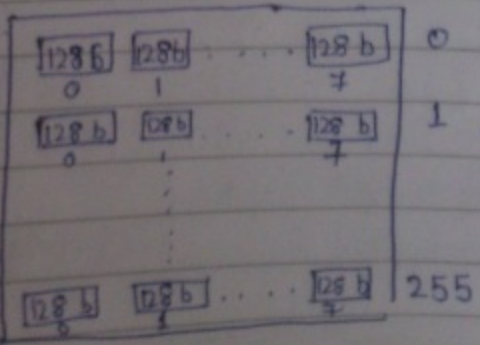
1 RAM Chip



16 KB

Q. RAM chip size = (128 words) × (1-bit) → each word of 1 bit
Mem. cap. = (32 K words) (B) → each word of 1 byte

$$\frac{2^5 \times 2^{10} \times 2^3}{2^7} = \frac{2^{18}}{2^7} = 2^{11} = \boxed{2048\ \text{RAM chips}}$$



← 8 × 256

Cosmos

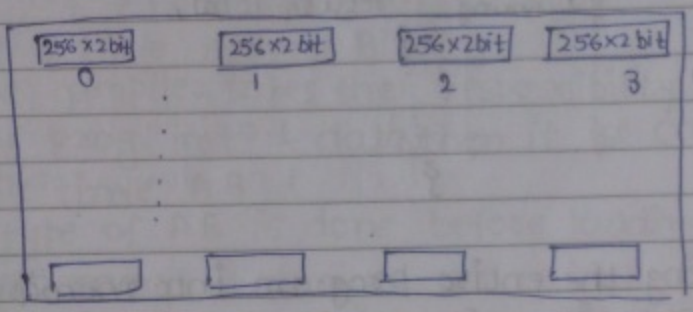
Q. RAM chip size = (256 words) × (2 bit) → each word of 2 bit each

Capacity = (64 MB) → each word of 1 Byte each

64 M words

No. of RAM chips = $\frac{2^6 \times 2^{20} \times 2^3}{2^8 \times 2} = \frac{2^{29}}{2^9} = 2^{20}$

1 Million RAM chips.

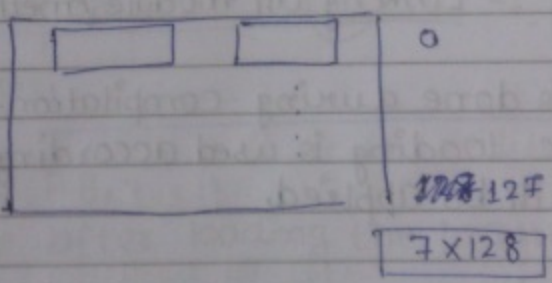


$\frac{2^{26}B}{2^8} = 2^{18}$

$2^{18} \times 2^8 = 2^{26}B$

Page no. 54

Q38. no. of RAM chips = $\frac{2^5 \times 2^{10} \times 8^7}{2^8 \times 4} = \frac{2^{18}}{2^8} = 2^8$



7 × 128

Date

Cosmos

10.06.12

Loading & Linking :-

Static dynamic

```
main()                    int add()
{ int x;                  {
  x = add();  ---linking--->  return total;
  ...                     }
  f()
  {
  }
```

Static :- Loading the entire program into memory before the starting of execution.

- Static loading suffers from inefficient utilization of memory.
- The program execution will be faster.

Linking :- Linking all modules/methods of the program.

- Linking is done during compilation.
- If static loading is used accordingly, the static linking will be applied.

Dynamic :-

- Loading the program into memory on demand (whenever it is required.)
- Efficient utilization of memory.
- Program execution will be slower.
- If the Dynamic loading is used acc., the dynamic linking will be applied.

Cosmos

- The majority of the OS uses dynamic loading & dynamic linking concept.

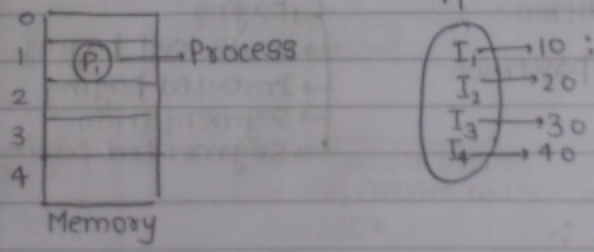
Address Binding :-

def^n :- Association of program inst^n & data to the actual physical mem. locations is called as address binding.

There are 3 types of address binding :-

① Compile time Address Binding :-

- If the compiler takes the responsibility of association of prog. inst^n & data, then it is called as compile time A.B.
- This type of A.B. is done before loading the prog. into memory.
- The compiler needs to interact with O.S memory manager to perform compile time A.B.



② Load Time A.B. :-

- It is done after loading the program into main memory.
- This type of A.B. will be done by the OS memory manager (loader).

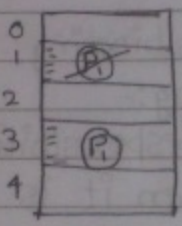
③ Run time A.B. :-

- The A.B. is postponed till even after loading the prog. in the main mem.

Cosmos

- This is done at the time of exec^n.
- The A.B. is done by processor (or CPU).
- Majority of the OS follows Run time or Dynamic A.B.

for run time



When the program is moved from block 1 to block 3 during run time due to paging, etc., the corresponding to block 1 we have diff. binded addresses of the inst^n than from block 3,
∴ A.B. is done during run time.

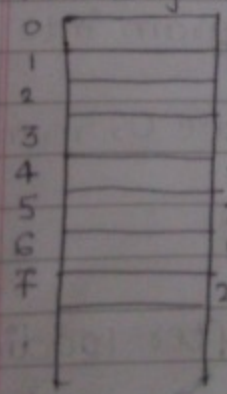
Memory Management Techniques:-

- Contiguous
  - fixed partition scheme
  - variable partition scheme
- Non-Contiguous
  - paging
  - multi-level paging
  - Inverted paging
  - segmentation
  - segmented paging

Contiguous :-

Fixed partition Scheme :-

- Memory is divided into fixed no. of partitions.



- No. of partitions are fixed, but not their size.
  eg. we have 8 partitions here, which are fixed.
- In every partition, we can only store 1 process,
- ∴ no. of processes are restricted by no. of partitions in the memory. (or degree of multiprog. is restricted by no. of partitions in mem.)

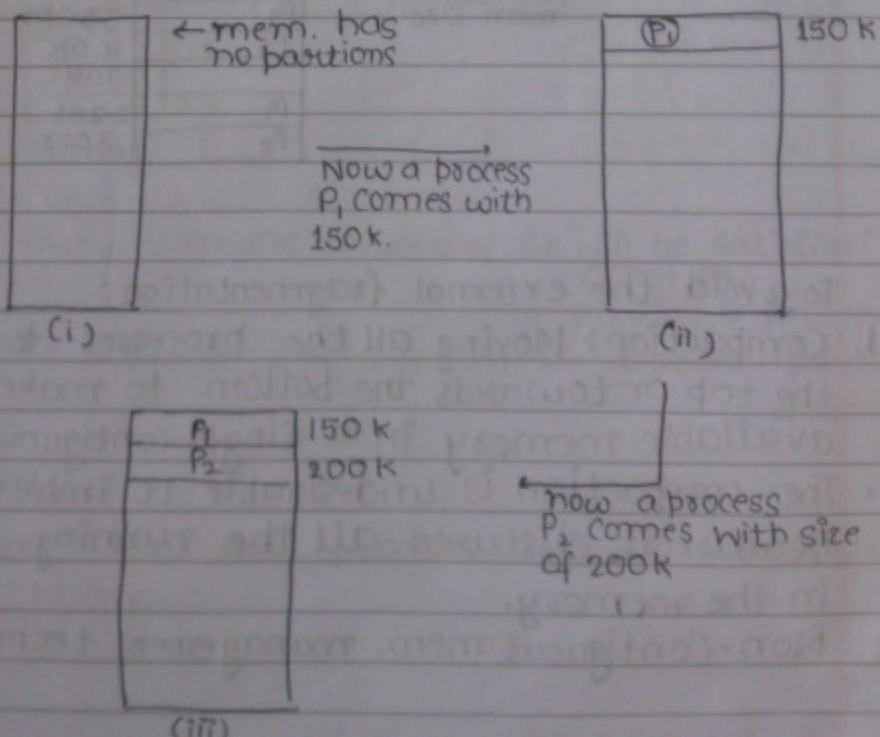
Cosmos

- The max. size of the process is restricted by max. size of the partition.
  e.g. we can't place 350K process in our example memory
- If we accomodate 20K process in partition of 30K, then 10K is wasted. This wastage of memory is called internal fragmentation.
- Every partition is associated with limit register, so lower limit contains starting address of the partition & upper limit contains ending address of partition.

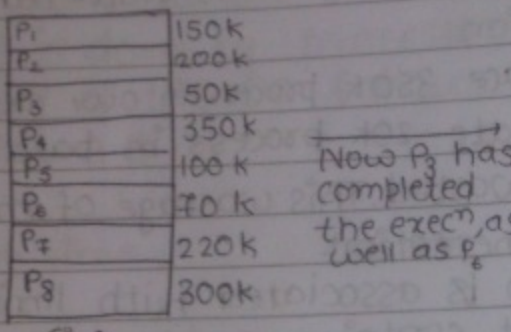
Variable Partition Scheme

- In variable Partition scheme, initially the memory will be full contigous free block.

So, initially:-



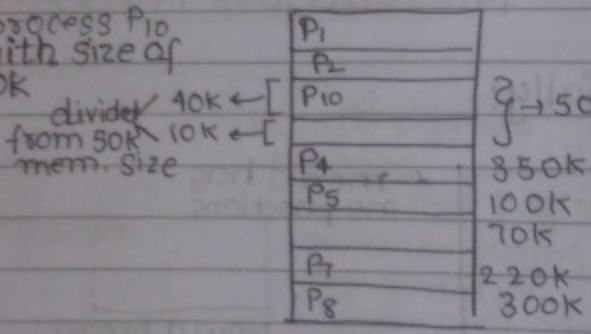
Cosmos



but this process P9 wont be allocated because we dont have 100K contiguous memory, this is called external fragmentation.

Now, if a process P9 comes with size of 100K [less than available free memory]

Now a process P10 comes with size of 40K



To avoid the external fragmentation:-

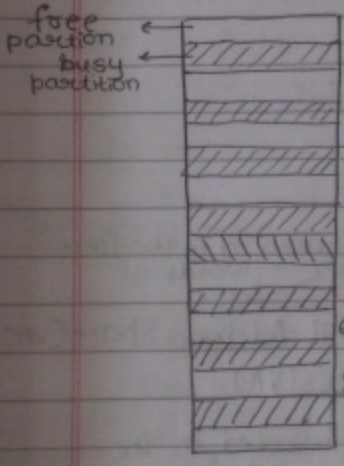
1. Compaction: Moving all the processes towards the top or towards the bottom to make free available memory in a single contiguous block.
   - The compaction is undesirable to implement because it disturbs all the running processes in the memory.
2. Non-Contiguous mem. management techniques.

Cosmos

Partition Allocation Techniques:-

1. If more than one partition is freely available t accomodate the process, then the decision making will be done with the help of partition allocatio techniques.

- First Fit :- Allocate the process in a partition which is 1st sufficie partition from the top.
- Best Fit :- Allocate the process in a partition which gives least inter fragmentation.
- Worst Fit :- Allocate the process in a partition which gives most interr fragmentation.
- Next Fit :- It also works like 1st fit, but it will search from last allocation point.

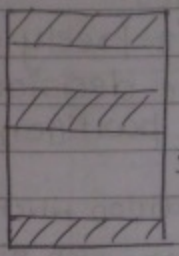


(i) P_1 comes with 16K
First Fit :- in 22k
Best Fit :- in 18k
Worst Fit :- in 36k
Next Fit :- in 36k (starting fro last allocation pt., the First fit for 16k i 36k)

Q. How many successive requests of 6K will be satisfied? (first fit).
(a) 17
(b) 18
(c) 19
(d) 20

[using variable length partition scheme
1 in 8k, 2 in 12k, 3 in 22k & so on.]

Q. Request from the processes are 300k, 25k, 125k, 50k respec.



The above request could be satisfied with: (Variable partition scheme)

Best fit:-

Ans.:- First Fit but not Best Fit

Non-Contiguous :-

Paging :-

1. Logical Address Space (LAS) or Virtual Address Space (Represented in the form of words / bytes)
2. Logical Address (LA) or Virtual Address (VA)
3. Physical Address Space (PAS) — represented in the form of bits.
4. Physical Address (PA) — represented in the form of bits.

represented in the form of words or bytes

e.g. • LA = 39 bits
LAS = 2^{39} words = 512 G words (if it is word addressable)

• LAS = 256 M words = $2^8 \times 2^{20}$ words = 2^{28} words.
∴ LA = 28 bits

• PA = 26 bits
PAS = 64 M words

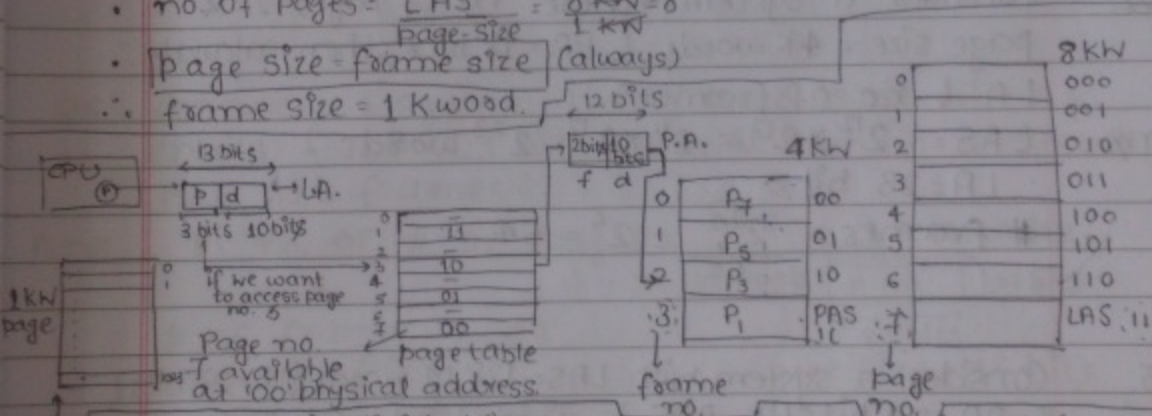
• PAS = 128 K words = 2^{17} words
∴ PA = 17 bits

Cosmos

★ Actual process is available in Physical Address Space
★ CPU generates Logical Address.
★ CPU can't directly access the process with the logical address generated by processor.
★ Maping Logical address to physical address is called paging technique.

let L.A. = 13 bits ∴ LAS = 8K words
& P.A. = 12 bits PAS = 4K words
} assuming word addressable memory

- page size = 1K words (assume)
- no. of pages = $\frac{LAS}{page\ size} = \frac{8\ KW}{1\ KW} = 8$
- page size = frame size (always)
- ∴ frame size = 1K word.



A page

- The LAS is divided into equal size pages.
- The PAS is divided into equal size frames.
- Some of the pages from LAS is brought into PAS.

★ No. of enteries in a page table = no. of pages in the LAS
- The page table entry contains the frame no.

★ p = no. of bits req. to represent the pages of LAS or page no.

★ d :- to represent particular words in a page. or no. of bits req to represent the page size of LAS or word no. of the page or page offset.

★ f :- no. of bits req. to represent the frames of PAS or frame no.

→ so, f will tell the frame no. & d is the frame offset in that particular frame.

Cosmos

Q1. Consider a system with LA = 27 bits & Physical Address = 21 bits & page size = 4K word, calculate no. of pages & no. of frames.

Ans. # pages = $\frac{2^{27} \text{ words}}{2^{12} \text{ words (in 1 page)}} = 2^{15} = 32K$

frames = $\frac{2^{21} \text{ words}}{2^{12} \text{ words (in 1 frame)}} = 2^9 = 512$

Q2. Consider a system where no of pages = 2K & page size = 4K words & PA = 18 bits, then calculate LA & no. of frames?

Ans. LAS = $2^{12} \times 2^{13} = 2^{11} \times 2^{12} = 2^{23}$ words

∴ LA = 23 bits

frames = $\frac{2^{18}}{2^{12}} = 2^6 = 64$

Q5. Consider a system with LAS = 128 M words [LA = 27 bi & PA = 24 bits] the PAS is divided into 8K frames. What is the page size & how many pages in the LAS?

Ans. page size = (or frame size) $\frac{2^{24}}{2^{13}} = 2^{11} = 2K$ wor

(total PAS / no. of frames)

pages = $\frac{2^{27}}{2^{11}} = 2^{16} = 64K$

Q4. Consider a system with LA = 32 bits & PAS = 64MB ($PA = 2^{26}$), page size = 4 kB, the mem. is byte addressable, the page table entry size is 2 bytes, what is the approx. page table size in bytes

(a) 1 MB (c) 4 MB

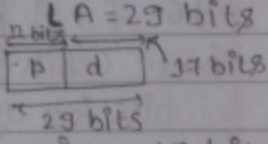
(b) 2 MB (d) 8 MB

Ans. page table entry size = 16 bits
∴ # of pages = 2^{16}
LA = 32 bits
LAS = 2^{32} Bytes
# of frames =
# of pages = $\frac{2^{32}}{2^{12}} = 2^{20} = 1M$
∴ page table size = 2B × 1M = 2MB

no. of occupied page table entries = no. of frames (max.)

Q5. Consider a system having page table with 4K entries & LA = 29 bits, what is the PA if system has 512 frames?

Ans. # of pages = 2^{12} (4K pages)

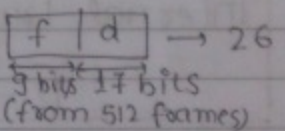


# of frames = 512

PA = PAS / # of frames

∴ PA = 26 bits.

∴ page size = 17 bits



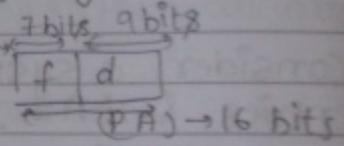
Q6. Consider a system with LAS = 256 MB [LA = 2^{28} bits] & PA = 22 bits, the PAS is divided into 8KB frames (page size). The mem. is byte addressable, what is the # of pages?

Ans # of pages = $\frac{2^{28}}{2^{13}} = 2^{15} = 32K$

Cosmos

Q. Consider a system with LAS = PAS = 2^{16} Bytes. The page size = 512 Bytes. The page table entry size is 2 Bytes. The mem. is byte addressable. The page table entry contains besides other info
1 bit → valid/invalid
1 bit → reference
3 bits → page protection
1 bit → dirty bit
How many bits are still available in the page table entry to store ageing info.

Ans. PA = LA = 16 bits.



# of frames = $\frac{2^{16}}{2^9} = 2^7$

∴ 7 bits are req. for frame no.
16 − 7 = 9 bits
but 6 bits for above info.
∴ bits for ageing = 9 − 6 = 3 bits.

★ Internal Fragmentation = $\frac{p}{2}$

p :- page size

Q Consider a system with LAS = PAS = S bytes. The page size = p bytes, page table entry size = e bytes. What is the optimal value of page size by minimizing the memory overhead of page table size + internal fragmentation in the paging.

(a) $P = \sqrt{2Se^2}$
(b) $P = \sqrt{2(Se)^2}$
(c) $P = \sqrt{2Se}$
(d) $P = \sqrt{2S^2e}$

* All the page table of the processes are placed in the main memory.

Ans. LAS = PAS = S bytes

# of pages =

# of frames = $\frac{S}{p}$ = # of pages

page table entry size = e bytes

So, the total size of page table = $\frac{S}{p} \cdot e$ bytes.

memory overhead = page table size + Internal frag.

$= \frac{S}{p} \cdot e + \frac{p}{2}$

$\frac{d}{dp}\left(\frac{S}{p} \cdot e + \frac{p}{2}\right) = 0$

$Se.\log p' + \frac{1}{2} = 0$ $\quad Se\left(-\frac{1}{p^2}\right) + \frac{1}{2} = 0$

$Se \log p = -\frac{1}{2}$ $\quad 2Se = p^2$

$\Rightarrow p = \sqrt{2Se}$

Q. * Main mem. access time = 'm'

If page tables are kept in main mem.

Effective memory Access time = $\underset{\text{to access page table}}{m} + \underset{\text{to access the page}}{m} = 2m$

If TLB is added to improve the performance The TLB contains the frequently refered page nos. & corresponding frame nos., the TLB is implemented with the help of associative register

TLB access time = 'c'

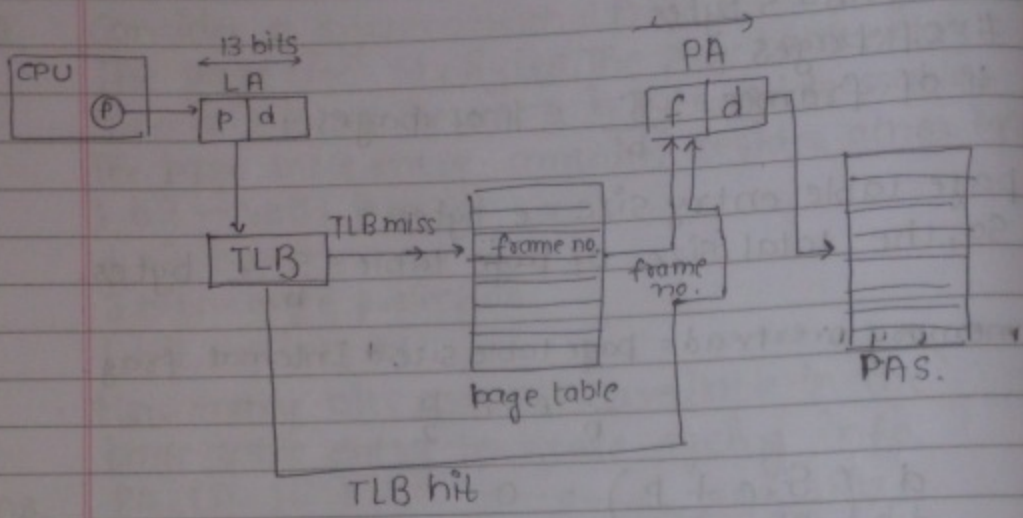
TLB hit ration = 'x'

∴ Effective mem. access time =

$[x(c + m)] + [(1-x)(c + m + m)]$

$x(c+m)$: when hit — TLB access, PAS access

$(1-x)(c+m+m)$: when miss — TLB access, page table access, PAS access



Q. Consider a system with main mem. access time 100 ns & TLB access time = 20 ns & TLB hit ratio 95%, what is effec. mem. access time with & W/O TLB.

Ans. W/O TLB = 2xm = 200 ns

With TLB = $0.95 \times (20+100) + 0.05 \times (20+100+100)$

$= 0.95 \times 120 + 0.05 \times 220$

$= 114 + 11$

$= 125$ ns.

Q. What hit ratio is req. to reduce effec. mem. access time from 300 ns (W/O TLB) to 250 ns (with TLB), the TLB access time = 60

Ans $2m = 300$ ns

$m = 150$ ns

$250 = x.(210) + (1-x)[360]$

$250 = 210x + 360 - 360x$

$110 = 150x$

$x = 11/15 = 0.73 \approx 0.73$

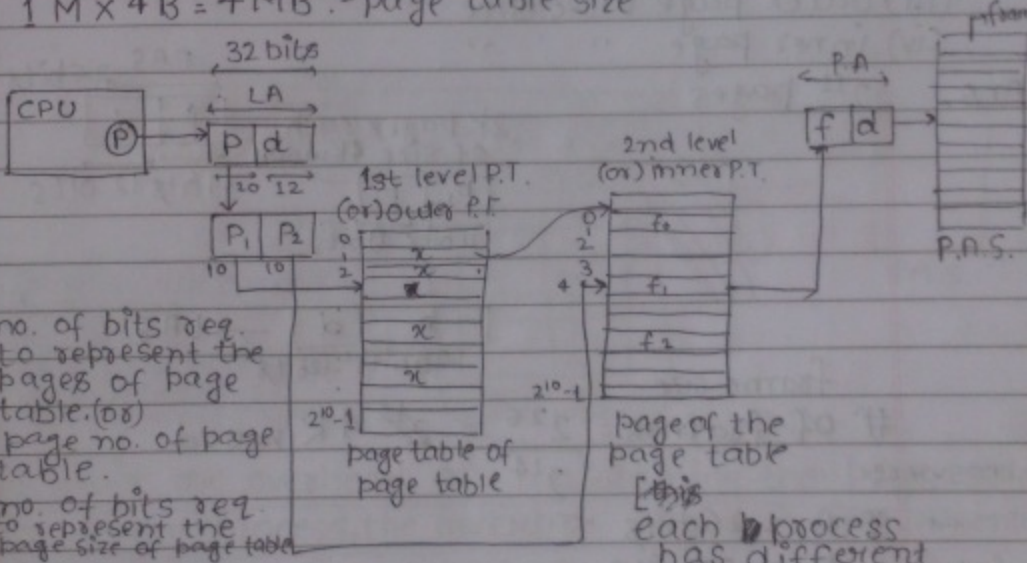
★ Every process has its own page table. To avoid the overhead of maintaining the large page tables, the multilevel paging will be implemented.

Multilevel Paging

Q. Consider a system with LA = 32 bits & page size = 4K words. The page table entry size = 4B, what is the page table size?

Ans # of pages = $\frac{2^{32}}{2^{12}} = 2^{20} = 1M$ ▶

$1M \times 4B = 4MB$:- page table size

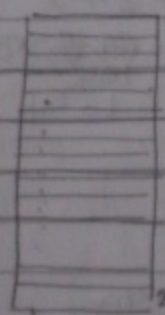


P1:- no. of bits req. to represent the pages of page table. (or) page no. of page table.

P2:- no. of bits req to represent the page size of page table

[this each process has different inner page table.

x:- address of page of page table
page size of page table = 1KW



single level paging :-

We are dividing the page table with page size of 1K

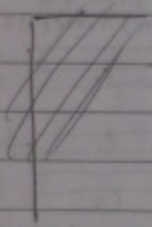
2^{20}

# of pages in page table = $\frac{2^{20}}{2^{10}} = 2^{10}$ pages in page table.

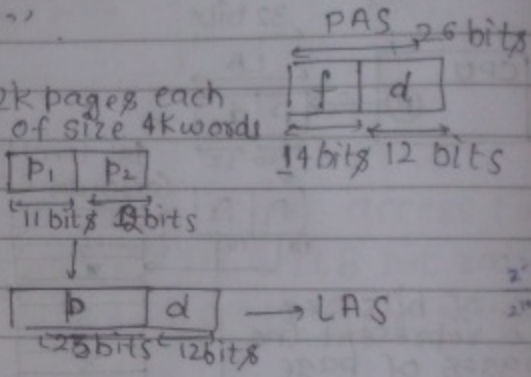
Q. Consider a system with 2 level paging applicable the page table is divided into 2K pages each of size 4K words, if PAS is 64M words which is divided into 16K frames. The page table entry size is 4B, then calculate

(i) length of L.A
(ii) length of P.A.
(iii) outer page table size
(iv) inner page " "

Ans. 2^{11} pages



frame size



# of frames = $\frac{2^{26}}{2^{14}} = 2^{12}$ 4K word

My answers:-

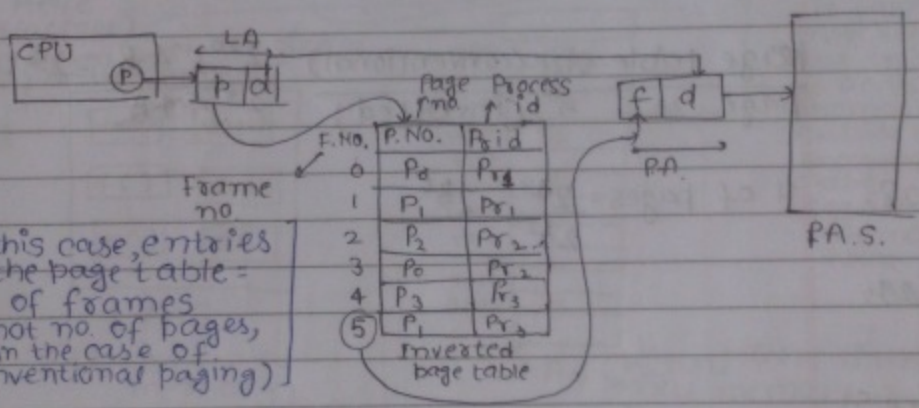
→ P.A = 26 bits
→ L.A = 35 bits
→ outer page table size = $2^2 \times 2^{11} = 2^{13}$ 8K
→ inner page table size = $2^{12} \times 2^2 = 2^{14}$ = 16K B

★ Performance of 2-level paging :-
Main mem. access time = 'm'
If the P.T.'s are kept in main mem.
Effec. main mem. = 3M

Cosmos

If TLB is added,
the performance :-
TLB access time = 'c'
TLB hit ratio = 'x'
$\therefore E_{MAT} = \boxed{x(c+m) + (1-x)(c+3m)}$

Inverted Paging



In this case, entries in the page table = no. of frames (& not no. of pages, as in the case of conventional paging)

★ To avoid the overhead of maintaining the page table for every process, the inverted paging is implemented. In the inverted paging, only 1 page table is maintained for all the processes.

★ No. of entries in the page table is same as the no. of frames in the P.A.S.

Q Consider a system with LA = 34 bits & P.A = 29 bits & page size is 16 KB. The page table entry size = 8B. Calculate the page table size in conventional paging & inverted paging?

ans # of pages = $\frac{2^{34}}{2^{14}} = 2^{20}$ Page table size (in conventional) = 8B × 1M = 8MB

# of frames = $\frac{2^{29}}{2^{14}} = 2^{15}$ Page table size (in inverted) = 32K × 8B = 256

Cosmos

Page 53
Q 31

L.A. = 32 bits
Page size = 4 KB
P.A.S = 128 KB

conventional page table:-

# of pages $= \frac{2^{32}}{2^{12}} = 2^{20}$

inverted

# of frames $= \frac{2^{17}}{2^{12}} = 2^5$

$\frac{\text{page table size (conventional)}}{\text{page }\,''\,''\text{ (inverted)}} = \frac{2^{20} \times 4B}{2^5 \times 4B} = 2^{15} : 1$

Q28. # of pages $= \frac{2^{48}}{2^{12}} = 2^{36}$

Q29.

Page 51
Q5.

Page size = 4 KB
LA = 32 bit
PA = 30 bit

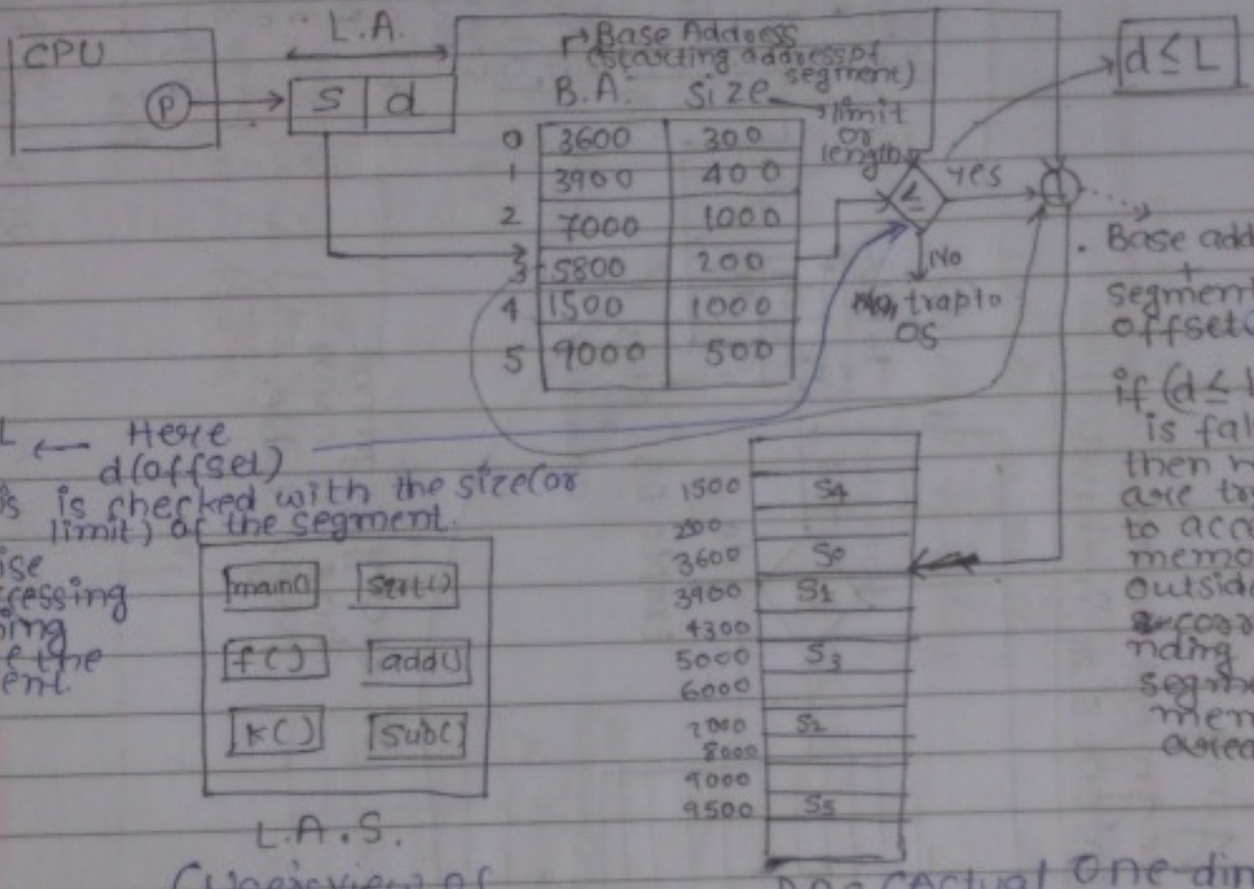
# of pages $= 2^{20}$

# of frames $= \frac{2^{30}}{2^{12}} = 2^{18}$

∴ page table size (inverted) $= 2^{18} \times (18 + 12)$ bits ← bits for no. of frames (= no. of pages in inverted page table)
$= 2^{18} \times (30)$ bits
$= 2^{23}$ bits
$= 2^{20}$ B

Cosmos

Segmentation



★ Paging doesn't follow user views of memory allocation. To achieve the user's view of memory allocation, the segmentation will be implemented. The L.A.S. is divided into various segments, the segments will vary in the size.

s:- no. of bits required to represent the segment of L.A.S.

d:- no. of bits req. to represent the size of segment or segment offset.

Segmented Paging

Paging on segment when segment size increases.

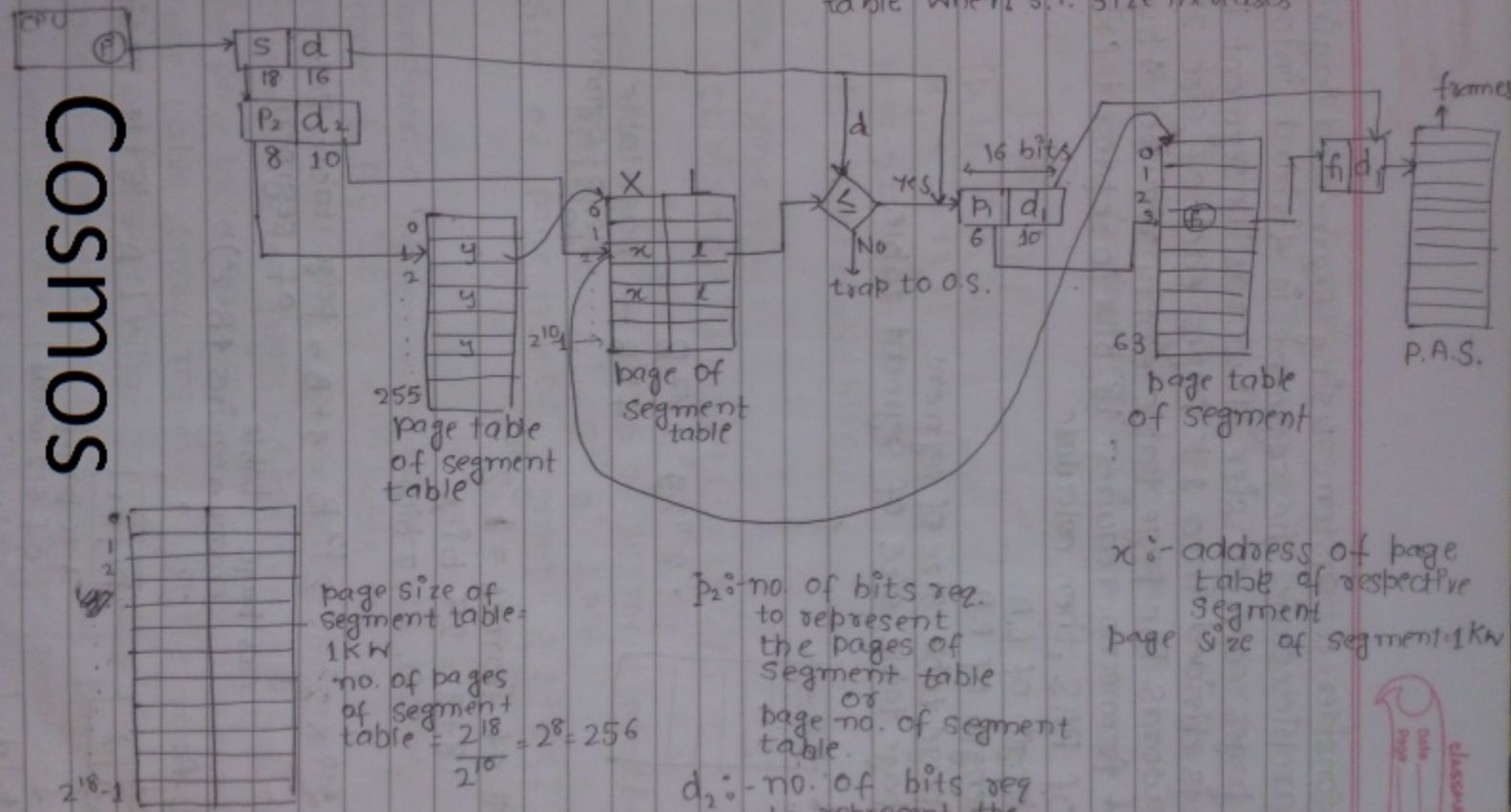
★ every segment has its own page table.

Paging on segment action

paging on segmentation

y :- address of page of segmented table
page size of segment table = 1 KW

paging of segment when segment size increases & also paging on segment table when S.T. size increases



page size of segment table = 1KW
no. of pages of segment table = $\frac{2^{18}}{2^{10}} = 2^8 = 256$

p_2 :- no. of bits req. to represent the pages of segment table or page no. of segment table.

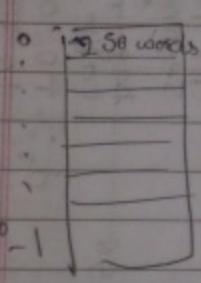
d_2 :- no. of bits req. to represent the page size of segment table.

x :- address of page table of respective segment
page size of segment = 1KW

Cosmos

Q. Consider a system using segmented paging architecture. The segment is divided into 1K pages each of size 512 words. The segment table is divided into 1 K pages each of size 256 words. The page table entry requires 4 bytes, & frame no. requires 18 bits to represent frames of P.A.S., then calculate

(i) length of L.A.
(ii) length of P.A
(iii) page table size of segment
(iv) page table size of segment table.

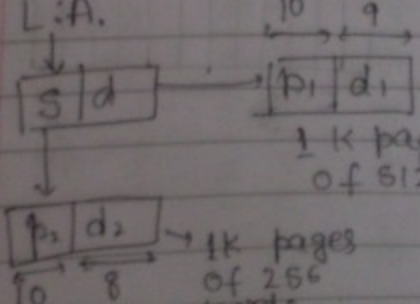


- $2^{10} \times 2^2 = 2^{12}$ B
  $= 4$ KB → page table size of segment table

- # of frames = 2^{18}
  18+9 = 27 bits
- ∴ P.A. = 27 bits.

- $2^{10} \times 2^2 = 2^{12}$ B = 4KB → page table size of segment.
  ↓
  (as page table entry requires 4B = 2^2)

- L.A.

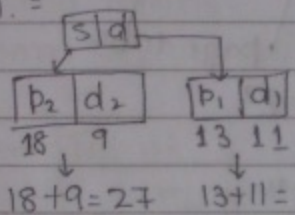


∴ L.A. = 18+19 = 37 bits

Cosmos

Q. Consider a segmented paging arch. where segment is divided into 8K pages each of size 2K words. The segment table is divided into 256 K pages each of size 512 words. The page table entry requires 64 bits of size & frame no. requires 22 bits to represent frames of P.A.S, then calculate

Ans. • L.A. =



18+9=27 13+11=24

∴ L.A. = 27 + 24 = 51 bits

• frame size =

22 bits + d, bits = 22 + 11 = 33 bits

now, size of page table of segment table =

$2^3 B \times 2^{18} = 2^{21} B = 2MB$

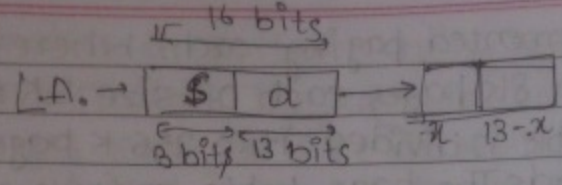
size of page table of segment = $2^3 B \times 2^{13}$

$= 2^{16} B = 64 KB$

Q. Consider a system with segmented paging arch. Where L.A.S. = P.A.S. = 2^{16} bytes, the L.A.S. is divide into 8 equal size segments, the segment is div into equal size pages which are of powers of 2 The page tables are stored in the main mem. A page table entry requires 2B. The memory is byte addressable. What must be the size of the page of segment in bytes, so that page table of segment exactly fits in 1 page frame.

Ans frame size = $\frac{2^{16}}{2^3} = 2^{13}$ bytes

Cosmos



$\frac{2^{16}}{2^3} = 2^{13}$ Bytes

1 segment is divided in page size which is power of 2

$\therefore 2^{13} B = n \times 2^x B$

$n = 2^{12}$ pages in page table

Let page size of segment $= 2^x$

frame size $= \frac{2^{16} \text{ Bytes}}{\text{no. of frames}}$

page size of segment = frame size of P.A.S.

$\therefore x^x$

page table size of segment = page size of segment

$\left(\frac{2^{13}}{2^x}\right) \times 2B = 2^x$

no. of entries in page table

each entry of 2B each.

$2^{14} B = 2^{2x} B$

$\therefore x = 7$

$\therefore 2^7 = 128B$

Date

Cosmos

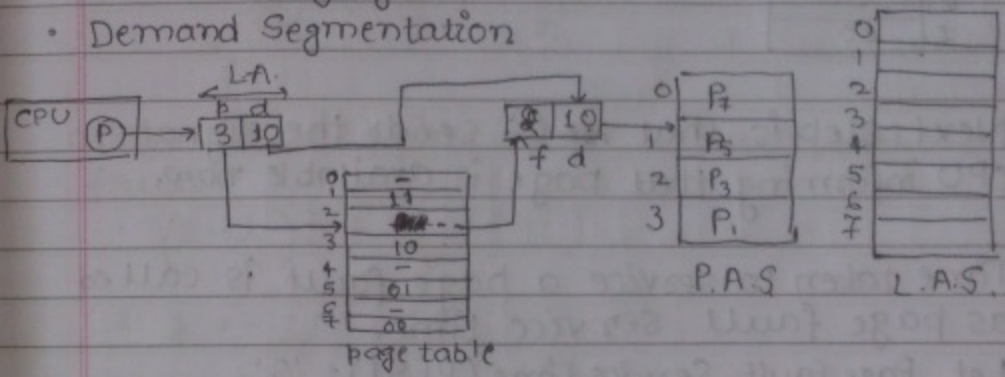
01.07.12

Virtual Memory

Def^n:- Virtual Memory gives an illusion to the programm that programs of larger size than actual physical memory can be executed.

It is implemented using:-

- Demand Paging
- Demand Segmentation



★ If we allocate all the pages of one process into physical mem., then less no. of processes can be allo to RAM, hence decreasing the degree of multiprogram

Page fault:- The processor is trying to access the page which not currently available in the main memory.

- When page fault occurs, then signal is sent to signalling processor requires a page currently present in the ~~phy~~ P.A.S.
- Now, OS validates the ~~pres~~ request of the proce
- Then, OS searches for the required page in L.A
- When OS found the req page, it brings the re page from L.A.S. to P.A.S.
- Let suppose, processor req. P_4 page not currently in PAS, brings P_4 in place of P_3 (assumption) & char the P.A.S. (we use page replacement algorithm for t
- Now req. page is in P.A.S., we now update the pa

table.

New page table:

0	–
1	11
2	–
3	10 –
4	10
5	01
6	–
7	00

- Next step is that OS sends the signal to CPU informing that page is available now.

★ Time taken to service a page fault is called as page fault service time.

Let Page Fault Service Time (PFST) = 'S'
& Main mem. access Time (MMAT) = 'M'
& Page Fault Rate = 'P'

Effective memory access time =

$(1-P) \times M + P \times S$

- $(1-P) \times M$: only one M & not 2M, because we ignore the page table access time
- $P \times S$: actually $(S + 3M)$
  - $M + M + M$
    - M → page table access time
    - M → main mem access time
    - M → page table access time (2nd time)
  - S → when there occur page fault (1st time)

hit ratio

Q. Consider a system with page fault rate of 99% & PFST is 10msec. & main mem. access time is 1msec, What is effec mem: access time

Ans. EMAT = $0.99 \times 1 + (0.01) \times 10$
$= 0.99 + 0.1 = 1.09$ msec.

Cosmos

Q. Suppose if an inst^n takes 'i' μsec & additional 'j' μsec. if page fault occurs. If a page fault occurs on an avg. for every k inst^n, then what is the effec. inst^n access time ?

Ans. $\left(1-\frac{k}{n}\right)i + \frac{k}{n}(j+i)$

$= 1 - \frac{k}{n} + \frac{k}{n}j + \frac{k}{n}i$

$= 1 +$

Page fault → for every k inst^n, there is a page fault.

$\therefore \left(1-\frac{1}{k}\right)i + \frac{1}{k}(j+i)$

$= i - \frac{i}{k} + \frac{j}{k} + \frac{i}{k}$

$= i + \frac{j}{k}$

Q. Let the page fault service time be 10msec & mem. access time is 20ns. If 1 page fault is generated for every 10^6 mem. accesses, what is the effec. access time of mem.

Ans. $(1 - 10^{-6}) \times 20\text{ ns} + 10^{-6} \times 10 \times 10^6$

$= 20\text{ns} + 10\text{ns}$

$= \boxed{30\text{ ns}}$

Q

Cosmos

Q. Consider a demand page env. where it takes 8ms to service a page fault, if either an empty frame is available or replaced page is not to be modified & it takes 20 msec. if replaced page is modified. Assuming main mem. access time = 1m sec. & further assume the page to be replaced is modified 70% of the time, what is the max. acceptable page fault rate to get effec. mem. access time not more than 2msec.

Ans. $0.7 \times p \times 20 + 0.3 \times p \times 8 + (1-p) \times 1 < 2$

$14p + 2.4p + 1 - p < 2$

$15.4p < 1$

$p < \frac{1}{15.4}$

$p < 0.06493$

(when main mem. access time is neglected when page fault occurs)

or

$0.7 \times p \times 21 + 0.3 \times p \times 9 + (1-p) \times 1 < 2$

$14.7p + 2.7p + 1 - p < 2$

$16.4p < 1$

$p < 0.06097$

(when main mem. access time is not neglected when page fault occurs)

★ Page modified means:-
when page is modified in P.A.S., & not in L.A.S., then the page is first saved in L.A.S. & then replaced.

Cosmos

Q. Consider a system where page fault service time = 200 msec. & main mem. access time = 10 msec., the TLB is added to improve the performance & 80% references are available in TLB & that of remaining 10% causes page faults, the TLB access time is negligible, what is the effec. memory access time?

Ans. $0.8 \times 10 + 0.2 \times (10(0.9) + 0.1 \times 200)$ (TLB hit, TLB Miss)

$= 8 + 0.2(9 + 20)$

$= 8 + 0.2(29)$

$= 8 + 5.8$

$= 13.8$ ms

Out of the 20% TLB Miss, 10% is causing page fault.

$0.8 \times 10 + 0.2 (0.1 \times 10 + 0.1 \times 200)$

$= 8 + 0.2 (1 + 21)$

$= 8 + 4.2$

$= 12.2$

Page Replacement Algorithm :-

Reference String :-

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

↳ page number referred by the process

Assumption :- The frames allocated to the process = 4.

*:- page fault occurs when #

FIFO (First In First Out)

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
		1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	7	7	7
7	7	7	7	7	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2
*	*	*	*	*	*	*	*					*	*			*			

raw # of frames: 3

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	1	1
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
*	*	*	*			*	*	*	*	*			*	*			*	*	*

Now, reference string:-

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

# of frames = 3

1	2	3	4	1	2	5	1	2	3	4	5
		3	3	3	2	2	2	2	2	4	4
	2	2	2	1	1	1	1	1	3	3	3
1	1	1	4	4	4	5	5	5	5	5	5
*	*	*	*	*	*	*			*	*	

# of frames = 4

1	2	3	4	1	2	5	1	2	3	4	5
			4	4	4	4	4	4	3	3	3
		3	3	3	3	3	3	2	2	2	2
	2	2	2	2	2	2	1	1	1	1	5
1	1	1	1	1	1	5	5	5	5	4	4
*	*	*	*			*	*	*	*	*	*

Cosmos

classmate
Date
Page

Belady's Anamoly:-

If the no. of frames allocated to the process increases the page fault increases.

Optimal Page Replacement:-

In the event of page fault, replace the page which is not used for longest duration of time in future.

# of frames=4.

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
4th				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3rd			1	1	1	1	4	4	4	4	4	4	4	4	4	4	7	7	7	
2nd		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1st	7	7	7	7	7	3	3	3	3	3	3	3	3	1	1	1	1	1	1	
↑ position	*	*	*	*		*		*						*			*			

7 is ~~used~~ not used for longest duration of time in future from 7, 0, 1, 2

here both 3 & 4 not used in future, now from 3 & 4, 3 comes 1st in FIFO because last replaced is from 3rd position, ∴ 3 → at 1st pos & 4 → at 3rd pos., we have to start from 4th pos. (1 plus from last replaced pos.)

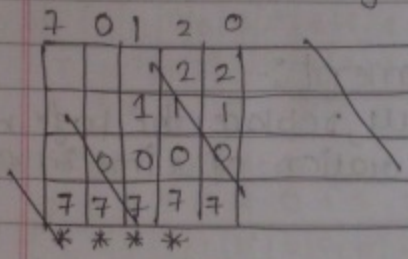
Same reason as of this

Cosmos

Least Recently Used :

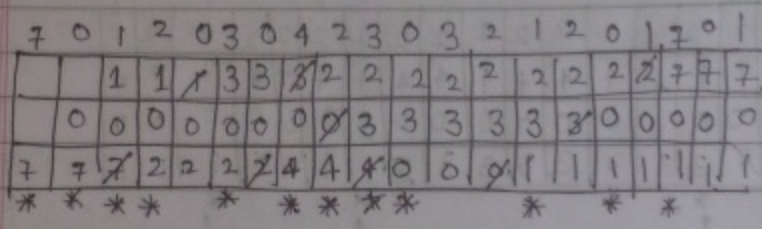
In the event of page-fault, replace the page which is ~~the~~ least recently used.

7 0 1 2 0



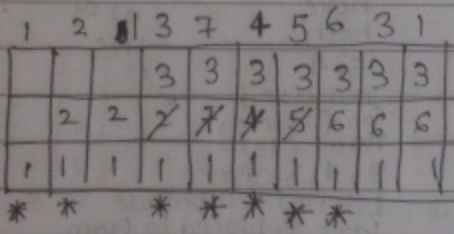
# of frames =3

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



Page 59
Q29.

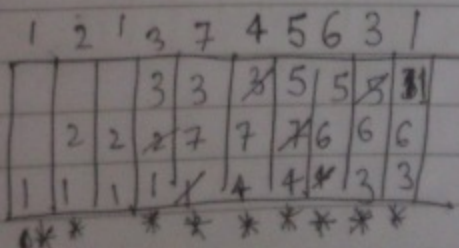
1 2 1 3 7 4 5 6 3 1



Q30.

1 2 1 3 7 4 5 6 3 1

1 2 1 3 7 4 5 6 3 1



Cosmos

Most Recently Used (MRU):-

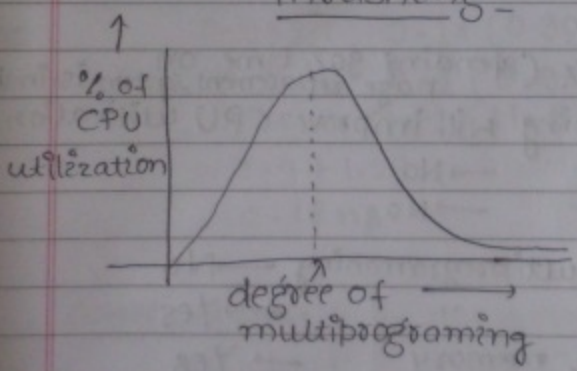
frames=4

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
				2	2	2	2	2	~~2~~	~~3~~	~~0~~	~~3~~	2	2	~~1~~	0	0	0	0	0
			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		0	0	0	~~0~~	~~3~~	~~0~~	4	4	4	4	4	4	4	4	4	4	4	4	4
	7	7	~~3~~	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
	*	*	**	*		*	*	*		*	*	*	*			*				

frames=3

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			~~1~~	2	2	2	2	2	~~2~~	~~3~~	~~0~~	~~3~~	~~2~~	~~1~~	~~2~~	0	1	1	1	1
		0	0	0	~~0~~	~~3~~	~~0~~	4	4	4	4	4	4	4	4	4	4	4	4	4
	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	~~7~~	0	0
	*	*	*	*		*	*	*		*	*	*	*	*	*	*	*	*		

Thrashing-



let free frames = 300
& no. of processes = 100,
then every process gets 3 frames each (with max. deg
of multiprogram

but if we have no. of processes = 300, then every proces
gets only 1 frame & page faults occur more
frequently & most of the time is spent

Cosmos

① Initially when the degree of multiprogramming $\leq \lambda$, the CPU utilization increases.
② If degree of multiprogramming is drastically increased after λ, then CPU utilization goes on decreasing.
③ This situation is called Thrashing.
④ In thrashing, the system will spend more time only on the page replacement.

Causes Of Thrashing :-

① High degree of multiprogramming.
② Lack of frames.

Q. Consider a system with the below statistics observed,
→ CPU utilization → 10%.
→ paging on disk → 90% (spending 90% time on page replacement, system is in thrash)
Which of the following will improve CPU utilization.

(a) Install faster CPU → No
(b) Install bigger Disk → No
(c) Increase degree of multiprogramming → No
(d) Decrease " " " → Yes
(e) Install more main memory → Yes
(f) Decrease page size → No (page fault because increases). no. of frames allocated to the pro are consider so decreasing page size causes the page fault to increase
(g) Increase page size → Yes likely

Cosmos

Q. The processor uses two-level page table for Virtual to Physical Address Translation. Page tables of both the levels are stored in the main mem. L.A = P.A. = 32 bits. The memory is byte-addressable. For Virtual to Physical Address translation the 10 most significant bits of V.A. are used as index into 1st level page table, while the next 10 bits are used as index into 2nd level page table. The 12 least significant bits of V.A. are used as offset within the page. Assume that page table entries in both the levels of page table are 4B wide. Further, the processor has TLB with hit ratio of 96%. The TLB caches recently used Virtual page nos. & corresponding Physical page nos. The processor also has a physically addressed cache with hit ratio of 90%, the main mem. access time = 10ns, cache access time = 1ns, TLB access time = 1 ns. What is effec. mem. access time?

My answers:-

Ans $0.9 \times 1 + 0.1 \times (0.96 \times 1 + 0.04 \times (20 + 10))$ (1+1)

$= 0.9 + 0.1 (0.96 \times 1 + 0.04 \times 32)$

$= 0.9 + 0.1 (11.52 + 1.28)$

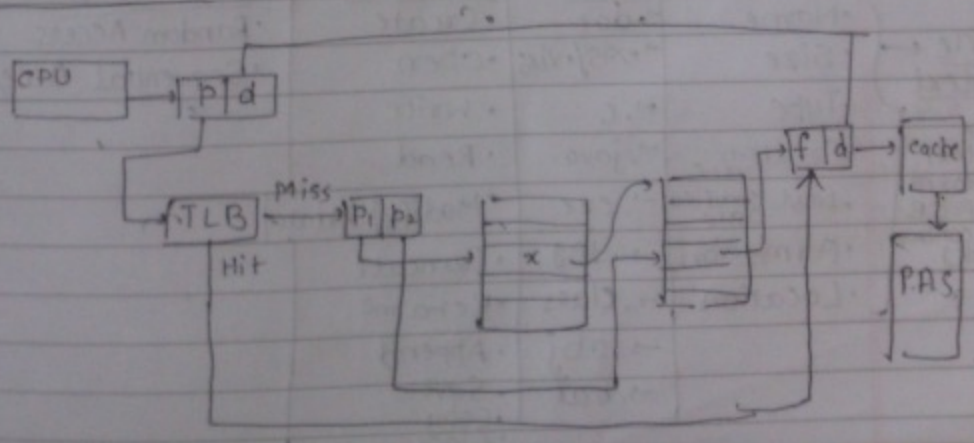
$= 0.9 + 1.28$

$= 2.18$ ns

21 × 9 = 189

→ wrong (correct only when Cache is logically addressed)

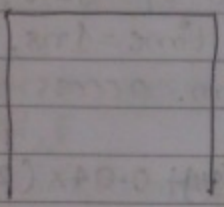
Correct Answer:-



$0.96(1+10)+0.04(0.9x$

$0.96(1+0.9x(1)+0.1x(1+10))+$
$0.04(1+10+10+0.9x(1)+0.1x(1+10))$
$=0.96(1+0.9+1.1)+$
$0.04(1+20+0.9+1.1)$
$=0.96(3)+0.04(23)$
$=2.88+0.92$
$=3.8$ ns.

Page no. 39 Q40.



File & Device Management

- Hard Disk
- Tape

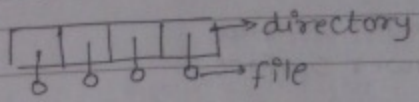
File :- Collection of logically related entities.

Attributes :-	Types :-	Operations :-	Access Methods :-
• Name	• doc	• Create	• Random Access
• Size	• .xls/.xlxs	• Open	• Sequential Access
• Type	• .c	• Write	
• Creation Date	• .java	• Read	
• Last Modified Date	• .exe	• Modify/Update	
• Permission	• .jpg	• Truncate	
• Location	• .class	• Rename	
	• .obj	• Append	
	• .bat	• Save	
		• Copy	
		• Close	

File context → File context is stored in F.C.B. (File control block).

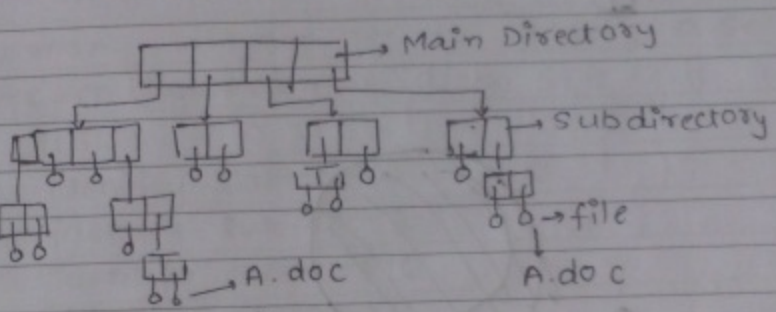
Directory Structure :-

(i) Single Level Directory :-



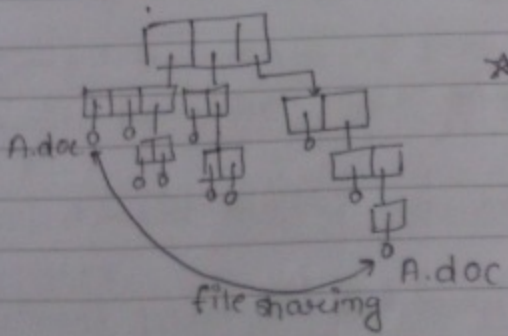
→ We can't have two files with same name.
→ Searching time of file will increase.

(ii) Tree Level Directory :-



→ Easy Classification
→ Less Search Time
→ If we change something in A.doc at one place, we explicitly need to perform same changes at other place.

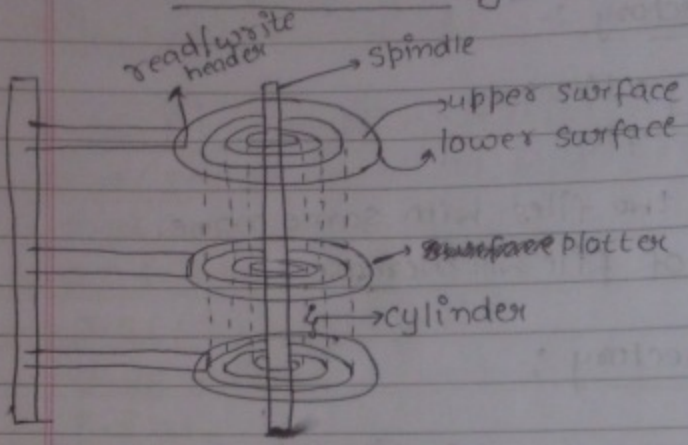
(iii) Acyclic directory Structure :-
(allows the file sharing)

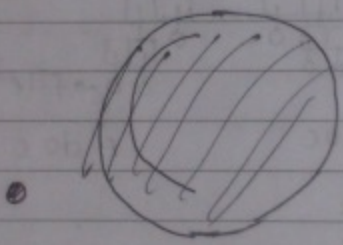


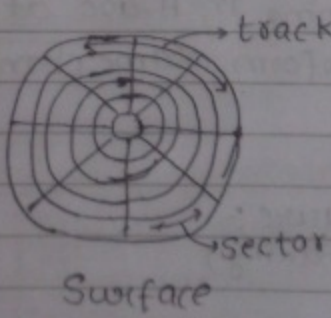
★ In this structure, if n changes A.doc at on place, then it will b modified same at t other place itself.

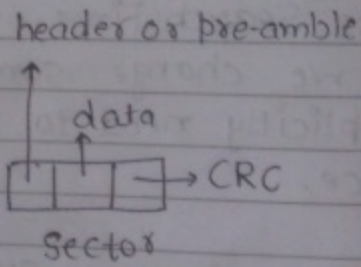
Cosmos

Disk Scheduling









Cosmos

Q Consider a disk with 16 plotters, each plotter is divided into 2 surfaces, every surface has 1K tracks & every track has 512 sectors & every sector can store 2 KB data, what is the capacity of disk?

Ans. $16 \times 2 \times 2^{10} \times 2^{9} \times 2 \times 2^{10}$ B

$= 2^{5} \times 2^{10} \times 2^{9} \times 2^{11}$ B

$= 2^{35}$ B

$= 32$ GB

Q. How many bits are req. to identify a sector in the above configuration of disk

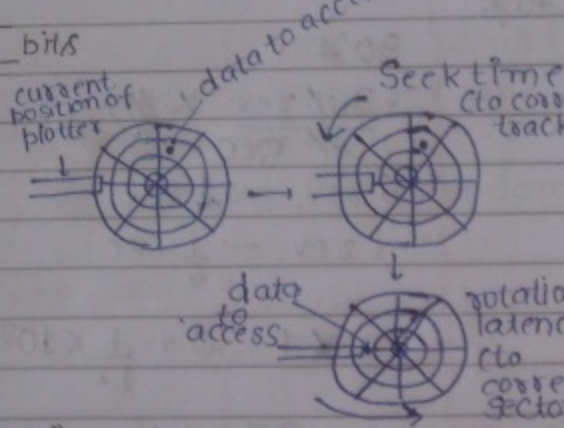
Ans. 24 bits.

to identify plotter → 4
to identify surface → 1
to identify track → 10
to identify sector → 9

24 bits

Disk I/O Operation :-

- Seek Time
- Rotational Latency
- Transfer time
- Transfer rate



- Seek Time :- The amount of time taken to move the read/write head from its current position to desired track.
- The read/write header can only can't be outside the plotter.
- Rotational Latency :- The amount of time taken to rotate the track when the read/write head comes

to exact position (exact sector within a track).
It is considered as $\frac{1}{2}$ x rotation time.

- Transfer Time :- The amount of time taken to transfer the required data. The transfer time depends on the total size of the track & rotational rate of the disk.

- Transfer Rate :- The no. of bytes transferred for unit time is called as transfer rate.

Q. If a disk system has an avg. seek time of 30 ns & rotational rate of 360 rpm. Each track of the disk has 512 sectors each of size 512 B. What is the time taken to read 4 successive sectors & also compute data transfer rate of the disk.

Ans.

30 ns

360 rev. / 60 sec.

6 rev. → 1 sec.

1 rev. → $\frac{1}{6}$ sec.

512 sector → $\frac{1}{6}$ sec.

1 sector → $\frac{1}{6 \times 2^9}$ sec

4 sector → $\frac{1}{6 \times 2^9} \times 4 = \frac{2}{3 \times 2^9}$

$30 + \frac{1}{12} \times 10^9 + \frac{2}{3 \times 2^9}$

30 ns

$= 30 \times 10^{-9} + \frac{1}{12} + \frac{2}{3 \times 2^9}$

negligible

≈ 0.0833 sec.

negligible

Transfer rate

Cosmos

Transfer time:-
In one rotation time → can read entire track data
time required? ← to read the req. data

1 rotatation time:-
360 → 60 sec:
1 → $\frac{60}{360}$ sec → $\frac{1}{6}$ sec.

Entire track size:-
512 × 512 B = $2^9 \times 2^9$ B = 2^{18} B = 256 KB
256 KB → $\frac{1}{6}$ sec.
1 B → $\frac{1}{6 \times 256 \times 2^{10}}$
2 KB → $\frac{1}{6 \times 256 \times 2^{10}} \times 2 \times 2^{10}$ →

1 sec. → 6 rev.
1 sec. → 256 KB × 6
1536 KBPS

Q. If a disk system has an avg. seek time of 60 ms & rotational rate of 3600 rpm. Each track of the disk has 256 sectors each of size 2 KB.
(a) What is the approx. time req. to read 1200 random sectors?
(b) What is the data transfer rate?

ans. 60 ms +
$\left(60 \times 10^{-3} s + \frac{1}{60} s \times \frac{1}{2} + \frac{1}{60 \times 256}\right) \times 1200$
$(0.00833) \times 1200$
≈ 10 sec.

$\frac{60}{3600}$ r →
60 rev. → 1 sec.
1 rev. → $\frac{1}{60}$ sec.
1 rev. → $\frac{1}{60}$ sec.
256 sec → $\frac{1}{60}$ sec.
1 sector → $\frac{1}{60 \times 256}$ sec

1 rev. → $\frac{1}{60}$ sec.
256 sectors → $\frac{1}{60}$ sec.
1 sec. → 256 × 60 sectors
1 sec. → $2^8 \times 2^{11} \times 60$ B
30 MBPS ← 30720 KBPS ← $512 \times 2^{10} \times 60$ B = $2^{19} \times 60$ B = $2^{19} \times 60$ B

Cosmos

Q. An application loads 100 libraries at startup. Loa each library requires exactly 1 disk access. The seek time of the disk to a random location is 10 msec, the rotational speed of disk is 6000 rpm. If all 100 libraries are loaded from random locati on disk, how long does it take to load all libraries (the time to transfer data from disk block once head has been positioned at the start of the block may be neglected)?

Ans.

$100\left(10\times10^{-3}+\frac{1}{2\times100}\right)$ Transfer time is neglected.

600 rev. → 60
1 rev. → $\frac{1}{100}$

$100\left(10^{-2}+\frac{1}{2\times100}\right)$

$1 + 0.5 = \boxed{1.5 \text{ sec.}}$

Q. Consider the Disk with below specifications:-

(a) No. of surfaces = 8
(b) Inner diameter = 4 cm
(c) Outer diameter = 12 cm
(d) Inner track distance → 0.2 mm [distance b/w any two consecutive concentric tracks]
(e) No. of sectors per track → 20
(f) Sector size → 4 KB
(g) Rotational Rate → 3600 rpm

(a) What is the capacity of disk?
(i) 512 MB (ii) 128 MB (iii) 64 MB (iv) 14 B

(b) What is data transfer?

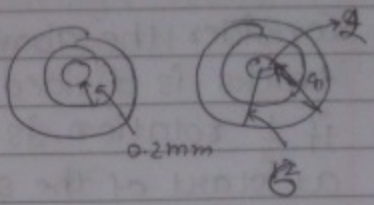
Cosmos

a) $8 \times 200 \times 20 \times 4$ KB

$8 \times 2 \times 8 \times 10^3$ KB

128×10^3 KB

128 MB



# of tracks $= \frac{4\text{ cm}}{0.2\text{ mm}} \times \frac{10\text{ mm}}{1\text{ cm}}$

$= \frac{400}{2} = 200$

360° rev. ⟶ 60 sec.

1 rev. ⟶ $\frac{1}{60}$ sec.

1 sec. ⟶ 60 rev.

1 sec. ⟶ 60 × 20 sectors

⟶ 60 × 20 × 4 KB

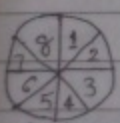
⟶ 1200 × 4 KB

⟶ 1.2 × 4 MB

⟶ 4.8 MB

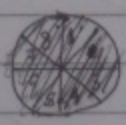
4.8 MBPS

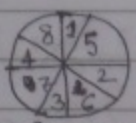
Disk Interleaving



No interleaving disk

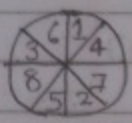
1 rotation





Single interleaving disk

2 rotations req. to read all 8 sectors of the disk



Double interleaving disk

3 rotations req. to read all 8 sectors of the disk.

(but exactly 2.75 rotations)

★ If there is slow

★ If it takes more time to read the data

Cosmos

Q. Consider the above double interleaving disk whose tracks is divided into 8 sectors each of size 2KB if $\frac{1}{2}$ rotation is req. to place read/write head at start of the sector, seek time = 30 ms & rotational rate is 3600 rpm.

(a) How much time is req. to read all 8 sectors

(b) What is the data transfer rate.

Ans. 360° rev. → 60° sec. 8 sector → $\frac{1}{60}$ sec

1 rev. → $\frac{1}{60}$ sec. 1 se

~~$\frac{1}{120}$~~ 3×10^{-2} s + $2.75\times\frac{1}{120}$ sec. + $8\ \frac{1}{60}\times2.75$.

My answer

$= 0.03 + 0.0229 + 0.0167$

$= 0.069$ sec. ✗ → Wrong.

(b) 1 sector → V

2.75 rev. → 8 sector

$2.75\times\frac{1}{60}$ sec. → 16 KB

349.1 KBPS ✓ → correct.

Cosmos

Sir's answer:- (a) Seek time + Rotational Latency + Transfer time
30 ms + 0.5 × Rotation time + 2.75 × rotation time

$= 30\text{ ms} + 0.5 \times \frac{1}{60} + \frac{2.75}{60}$

$= 30\text{ ms} + 8.33\text{ ms} + 45.83\text{ ms}$

$\approx 84\text{ ms}$

(b)