INDIA PUSTAK CENTRE

&

INDIA PHOTOSTAT

Name: BALAJI.SIR.....

Subject:.OPERATING.SYSTEM.....

Institute:...MADE.EASY..2020,2021.....

- Mechanical Engineering
- Civil Engineering
- Electronics Engineering
- Electrical Engineering
- Computer Science
- Instrumentation Notes

ALL CLASS NOTES & STUDY MATERIAL

IES, GATE, PSU's, BANK & RAILWAY

MRP.80

About Us

- Available SSC Class Notes for Civil, Mechanical and Electrical Branch.
- IES, GATE & PSU ' S, Class Notes Available for Mechanical, Civil, Electrical, Electronic, & Conventional.
- All Books Available for MADE EASY, ACE ACADEMY & IES MASTER.
- MADE EASY, ACE ACADEMY and IES MASTER Topper Class Notes.
- All Compitatid Exam Books and Notes Available for SSC ISRO, CGL, UPPSC, UPSC, Bank &Railway.

+91 956 012 7702, 9560814703

+91 956 0127702

Hindkhan777@gmail.com

WWW.indiapustak.com

Saket Metro, Near Made Easy and ACE Academy, Lane No. 4 Western Marg, Saidulajab
New Delhi-110030

wwwhindadda.com
9560127702

Operating Systems

Trapti Singh..

Teaching Schedule

I. Introduction and Background.

II. Process Management
- → process concept
- → CPU scheduling ✓
- → Synchronization ✓
- → Concurrent Programming.
- → Deadlocks
- → Threads.

III. Memory Management.
- → RAM Chip Implementation
- → Loading, Linking & Address Binding
- → Techniques
  - paging
  - Multilevel paging.
  - Inverted paging
  - Segmentation
  - Segmented Paging.
- → Virtual Memory.

IV. File Systems.

Textbooks

1. OS by Galvin.
2. Modern OS by A.S. Tenenbaumy.
3. OS by William Stallings.

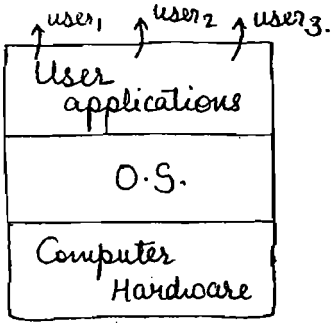
WWW.INDIANRUSTAK.COM 9560127702

Chapter 1

Introduction and Background

Q. What is an OS?

OS is an "interface between user and computer hardware".




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

→ internally calls write() System Call inorder to communicate with the monitor.

- System Call: System call is the request made by the user program to the OS in order to get any kind of service.
- Operating System is also called as Resorce Allocator because it is responsible for allocating resorces of a computer.

WWW.INDIAPUSTAK.com 9560912702

Resorces

- H/w Type — Eg. Devices, Memory.
- S/w Type — eg. Files, Directories.

Goals of O.S.

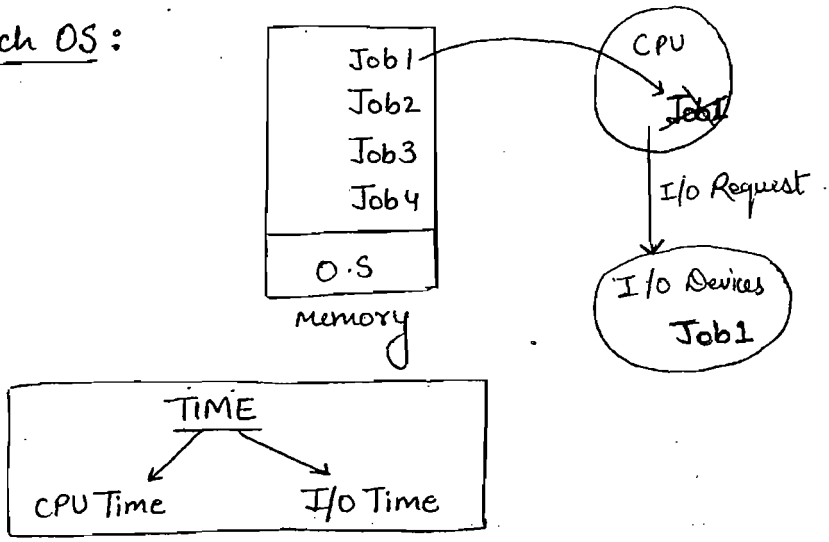
1. The primary goal is convenience. (easy to use)
2. The secondary goal is efficiency. (Stability).

~~Types of OS~~

~~1).~~

Types of OS

(1). The Batch OS :



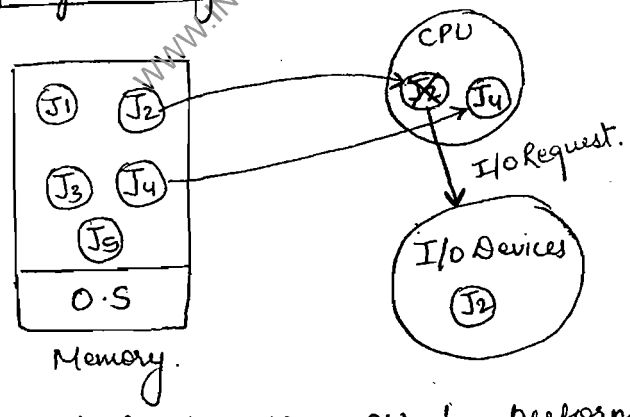
- If the Job is completed completely then only another Job will be scheduled onto CPU.
- increased CPU idleness.
- Decreased throughput of the system.

Throughput : No. of jobs completed per unit time is called throughput of the system.

Exp: IBM OS/2

WWW.INDIAPUSTAK.com 9560127702

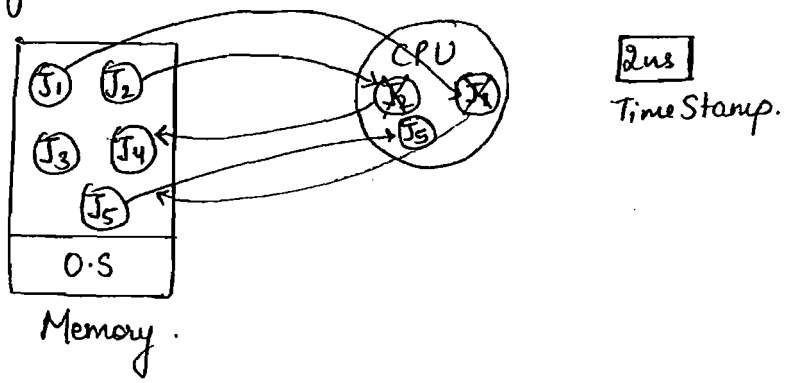
(2). Multiprogramming O.S. :



- If the job is leaving the CPU to perform IO operation, then another job which is ready for execution will be scheduled onto CPU.
- Advantage
  - Increased CPU Utilization.
  - Increased throughput of the system.

Exp : Windows, UNIX.

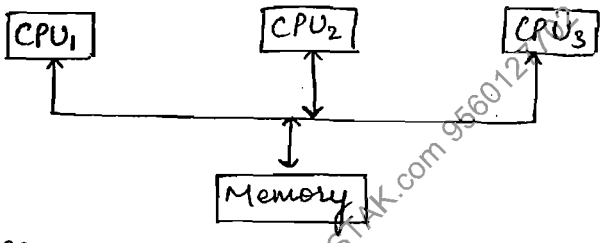
(3). Multitasking OS :



- Multitasking is an extension of multiprogramming OS.
- The jobs will be executed in the time sharing mode.

Exp: Windows, Unix

(4). Multiprocessor Systems :

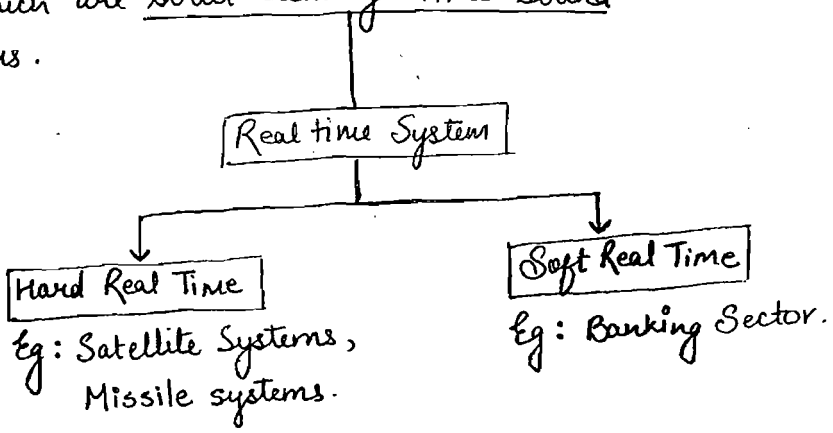


- Advantage
  - Increase the throughput of the system
  - Reliability
    - → Fault Tolerant Systems.
  - Economical.

Exp: UNIX.

(5). Real Time Systems :

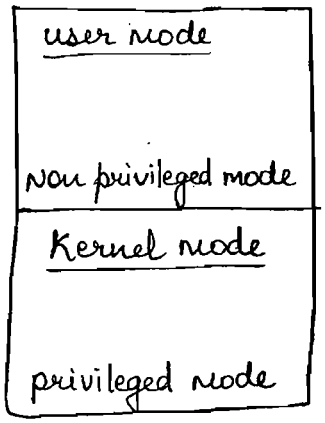
- The systems which are strict deadly time bound are called as real time systems.

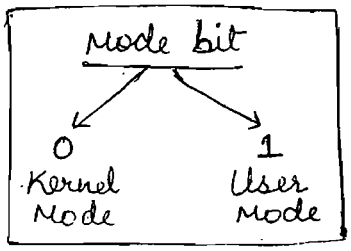


Exp: Sx works, Vx works, RTO's.

WWW.INDIAPUSHK.com 9560127878

Dual Mode Operation:





- In the hardware Level, the instructions are executed by using dual mode operation like
  1. user mode / Non privileged mode
  2. Kernel mode / priviledged mode / system mode / monitor mode.
- The dual mode operation is used in order to provide protection & security to the user programs. and also to the operating system from "errant users" (unauthorized users).
- It is purely the decision of the operating system in which particular mode, the instruction has to be executed.
- The mode bit is used to identify in which particular mode, the current instruction is executing.
- The privilised instructions are executed in the kernel mode & Non privilised instructions are executed in the user mode.
- In the Boot time, the system always starts only in the Kernel mode.
- The operating system always runs only in the kernel mode

Note: The mode switching takes very less time compared to process switching.

WWW.INDIAPOSTARK.com 9560127702

Privileged Instructions :

1. I/O operation
2. Context Switching
3. Disabling the interupts
4. Set the time of clock.
5. Clearing the memory map.
6. Changing the memory map.

Non-Privileged Instructions :

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

WWW.INDIAPUSTAK.com 9560127702

Privileged Instructions :

1. I/O operation
2. Content Switching
3. Disabling the interupts
4. Set the time of clock.
5. Clearing the memory map.
6. Changing the memory map.

Non-Privileged Instructions :

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

* FORK() System Call Implementation

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

- Fork is a system call used to create the child process.
- Fork returns -ve value, if the child process creation is unsuccessful.
- Fork returns value 0 to newly created child process.
- Fork returns +ve value (process id of child process) to the parent process.

WWW.INDIAPUSTAK.com 9560127702

Scanned by CamScanner

- When the child process is created, New memory location will be allocated to child process & the parent and child processes will have same virtual address but physical address will be different.

eg:

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

$\xrightarrow[\text{process}]{\text{it will create a child}}$

```
main()
{
  pf("Hello");
}C1
```

Hello Printed 2 times.

eg:

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

$\longrightarrow$

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

$\longrightarrow$

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

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

- 3 child process
- 4 times Hello printed.

WWW.INDIAPUSTAK.com 9560127702

eg:

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

$\longrightarrow$

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

$\longrightarrow$

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

$\longrightarrow$

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

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

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

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

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

- if the program is having n fork call then it will create $2^n - 1$ child.

Scanned by CamScanner

Chapter 2

Process Management

Process Concept

Def^n: The program under execution is called as process.

1). It should reside in the Main Memory.
2). It is occupied the CPU, to execute the instructions.
3). Active & Dynamic.

Attributes: Process is having various attributes.

1). Process ID
2). Process State
3). Program Counter
4). Priority
5). General Purpose registers.
6). List of open files.
7). List of open devices
8). Protection Information.

WWW.INDIAPUSTAK.com 9560127702

① Process ID : Process ID is the unique identification no. assigned by the OS at the time of process creation.

② Process State : Process state contains current state information of a process where it is residing.

③ Program Counter : Program Counter contains address of the Next instruction to be executed.

④ Priority : Priority is the parameter assigned by the OS at the time of the process creation.

* All the attributes of the process is ~~known~~ called as Context of the process.

* Context of the process will be stored in the PCB [Process Control Block]

Pr.ID	P.S
P.C	priority
GPR	LOF
L.O.D	P.I

P.C.B

- Every process will have its own P.C.B.
- P.C.B's will be stored in the Main Memory.
- P.C.B's will be implemented by using Double L.L.

States: Process will have different states.

1). New
2). Ready
3). Run
4). Wait or Block.
5). Termination or Completion.
6). Suspend Ready
7). Suspend Wait.

Operations: Various operations performed on the process.

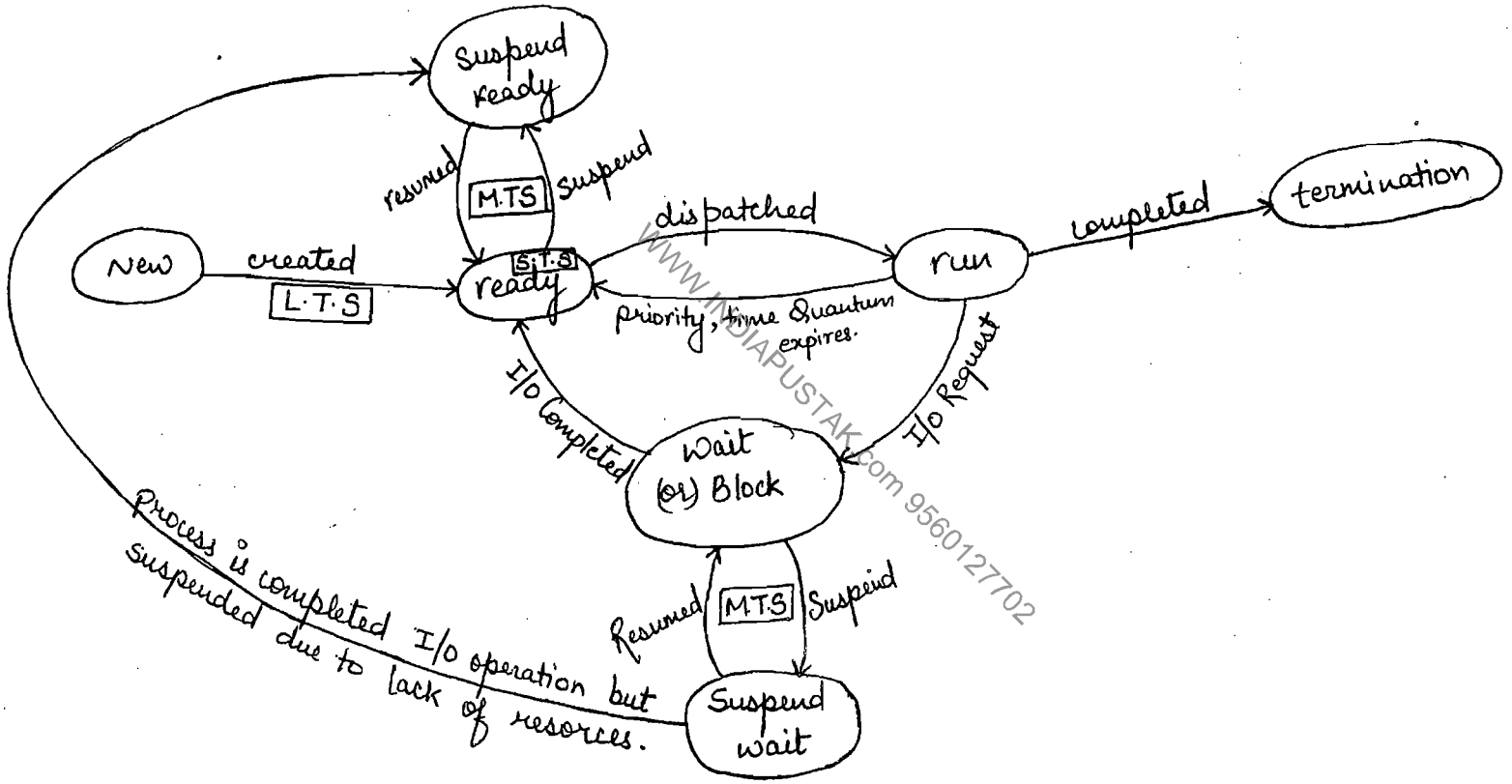
1). Create
2). Scheduling
3). Dispatching
4) Executing
5). Killing or Terminating
6). Suspending
7). Resuming.

WWW.INDIAPUSTAK.com 9560127702

PROCESS STATE DIAGRAM →

- Initially, process will be in the New state, it means process is under creation or process is being created.
- Once the process is created, it will be moved on to ready state and in the ready state there will be multiple No. of process.
- One of the process will be selected from the ready state and that will be dispatched onto the running state.
- When the process is in the running state, it is occupied the CPU and executes the instructions of the process and performing CPU time.
- In the running state, there will be only one process at any point of time.
- If the running process requires I/O operation than it will be moved onto wait or Block state.
- In the wait state also, there can be multiple number of processes. It means multiple process will perform I/O operation simultaneously.

Process State Diagram



WWW.INDIAPUSTAK.com 9560127702

- Multiprogramming
  - non-preEmptive
  - pre-Emptive. (or) Multitasking. (or) Time Sharing (time Quantum).

- When the process is in the ready state or in the running state or in the wait state, then it is residing in the main memory.
- When the resorces are Not sufficient to Manage the processes in the ready state then some of the processes will be suspended and they will be moved onto suspend ready state.
- When the process is in the suspend ready state, then it is residing in the Backing Store. (Secondary Memory)

WWW.INDIAPUSTAK.COM 9560127702

Degree of Multiprogramming

- The No. of processes present in the MM at any point of time is called as degree of multiprogramming.

Processes: Processes are categorised into 2 types:

1). CPU Bound 2). I/O Bound.

① CPU Bound: The processes which require more amount of CPU time are called as CPU bound processes.

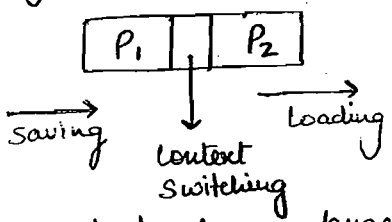
- These processes will spend more time on the running state.

② I/O Bound: The processes which require more amount of I/O time are called as I/O Bound process.

- These processes will spend more time in the wait (or) Block state.

- Each and every time, when the process will move from one state to another state, then context of the process will change.

Context Switching :



- Saving the context of one process and loading the context of another process is called as Context Switching.
- If the context of the process is more than context switching time will also increase which is undesirable.
- Context Switching time is considered as overhead for the system.

Note: In some special cases, if there is only one process still it is called as Context Switching.

Exp: Round Robin scheduling with 1 process.

WWW.INDIAPUSHK.COM 9560127702

In the O.S. there are 3 different Schedulers :

1). Long Term Scheduler (L.T.S) (Job Scheduler) : It is responsible of creating & bringing the new processes into the system.

2). Short Term Scheduler (S.T.S) (CPU Scheduler) : Short Term Scheduler is responsible for selecting one of the process in ready state for scheduling on to the CPU. (running state).

3). Mid Term Scheduler (M.T.S) : Mid Term Scheduler is responsible of suspending & resuming the processes.

- The job done by mid term scheduler is called as Swaping.

In computer, inside C drive type "*scheduler*" to see all the scheduler used.

Dispatcher: - Dispatcher is responsible for loading the selected job on to the CPU.

- It is also responsible for performing context switching.

- Long Term scheduler should select good combination of both CPU bound and IO bound processes inorder to get good throughput for the system.
- Long Term Schedules controls degree of multiprogramming.

Ques. Consider a system which has 'n' CPU processor's. then what is the minimum & maximum no. of processes that may present in ready, running & block states.

	min	max
Ready	0	Any.
RUN	0	n
Block	0	Any

"Any" depends on maximum degree of multiprogramming in the system.

WWW.INDIAPUSTAK.com 9560127702

19/11/2019

TIMES:

1). **Arrival Time:**
- The time when the process is arrived into ready state is called as arrival time of a process.
- It is also known as Submission Time.

2). **Completion Time:**
The time when the process is completed its total execution is called as completion time of a process.

3). **Burst Time:**
The time required by the process for its execution is called as Burst time of a process.

4). **Turn Around Time:**
The time difference b/w arrival time & completion time is called as turn around time of a process.

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

5). Waiting Time:

Time difference b/w Turn around time & Burst time is called as waiting time of a process.

$$\boxed{W.T = T.A.T - B.T}$$

6). Response Time:

The time difference b/w first response & arrival time is called as Response Time of a process.

$$\boxed{RT = \text{First Response} - AT}$$

CPU Scheduling

who: Short Term Scheduler.

where: Ready State.

when: 1. Run → Termination

2. Run → Wait

Run → Ready.

2. Wait → Ready

3. new → Ready.

WWW.INDIAPUSTAK.com 9560127702

Goal of CPU Scheduling

1. To maximise the CPU utilization & throughput of the system
2. To minimise the avg. TAT, avg WT and avg R.T. of the processes.

1. FIRST COME FIRST SERVE (FCFS)

Criteria – Arrival time

Mode – NON preEmptive.

Ques.

P.NO.	A.T.	B.T.
1	0	2
2.	1	3
3.	2	6
4.	3	5
5.	4	4

avg TAT = ?

avg WT = ?

Gantt Chart ↓

P_1	P_2	P_3	P_4	P_5

0 2 5 11 16 20

P.NO	AT	BT	CT	TAT	WT
1.	0	2	2	2	0
2.	1	3	5	4	1
3.	2	6	11	9	3
4.	3	5	16	13	8
5.	4	4	20	16	12
				44	24

$$\text{avg TAT} = \frac{44}{5} = 8.80$$

$$\text{avg WT} = \frac{24}{5} = 4.80$$

Ques2.

P.No.	AT	BT	CT	TAT	WT
1.	6	5	32	26	21
2.	3	7	18	15	8
3.	5	6	27	22	16
4.	1	2	3	2	0
5.	4	3	21	17	14
6.	2	8	11	9	1
				91	60

$$\text{avg TAT} = \frac{91}{6} = 15.16$$

$$\text{avg WT} = \frac{60}{6} = 10$$

Gantt Chart

	P_4	P_6	P_2	P_5	P_3	P_1

0 1 3 11 18 21 27 32

WWW.INDIANTUTOR.com 9560127702

Note: If the arrival times of the processes are matching then schedule the process which has lowest process ID.

Ques.3.

PNo	AT	BT	CT	TAT	WT
1.	3	1	6	3	2
2.	9	9	18	9	0
3.	12	8	26	14	6
4.	2	3	5	3	0
5.	15	7	33	18	11
6.	3	2	8	5	3
				52	22

$$\text{avg TAT} = \frac{52}{6} = 8.66$$

$$\text{avg WT} = \frac{22}{6} = 3.66.$$

Gantt Chart.

	P_4	P_1	P_6		P_2	P_3	P_5

0 2 5 6 8 9 18 26 33

Ques.

P.No	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	1	21	20	19
3	2	1	22	20	19
				60	38

$$\text{avg TAT} = \frac{60}{3} = 20$$

$$\text{avg WT} = \frac{38}{3} = 12.66$$

P_1	P_2	P_3

0 20 21 22

Ques.

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

avg WT = 0

avg TAT = $\frac{22}{3}$ = 7.33

P_1	P_2	P_3

0 1 2 22

Convoy Effect: In the FCFS, if the first process is CPU bound process & followed by many IO bound processes then it will have major effect on avg waiting time of the processes. This effect is called as the convoy effect.

2 SHORTEST JOB FIRST (S.J.F)

Criteria: burst time

Mode: NON preEmptive.

Ques.

PNO	AT	BT	CT	TAT	WT
1.	1	7	8	7	0
2.	2	3	14	12	9
3.	3	2	11	8	6
4.	4	1	9	5	4
5.	5	6	25	20	14
6.	6.	5.	19	13	8
				65	41

avg TAT = $\frac{65}{6}$ = 10.83

avg WT = $\frac{41}{6}$ = 6.83

	P_1	P_4	P_3	P_2	P_6	P_5

0 1 8 9 11 14 19 25.

Note: If the burst times of the processes are matching then, schedule the process which has lowest arrival time.

WWW.INDIAPUSTAK.com 9560127702

Ques.

P.NO	AT	BT	C.T	TAT	WT
1.	21	26	131	110	84
2.	17	19	82	65	46
3.	13	15	44	31	16
4.	2	27	29	27	0
5.	15	19	63	48	29
6.	19	23	105	86	63
				367	238

	P_4	P_3	P_5	P_2	P_6	P_1

0 2 29 44 63 82 105 131

$$\text{avg TAT} = \frac{367}{6} = 61.16$$

$$\text{avg WT} = \frac{238}{6} = 39.86$$

3. SHORTEST REMAINING TIME FIRST (S.R.T.F)

Criteria : Burst Time

Mode : preEmptive.

Ques.

P.NO	A.T	B.T	CT	TAT	WT
1.	0	~~7~~ ~~6~~ 0	21	21	14
2.	1	~~5~~ ~~4~~ 0	15	14	9
3.	2	~~3~~ ~~2~~ ~~1~~ 0	6	4	1
4.	3	~~1~~ 0	4	1	0
5.	4	~~2~~ 0	8	4	2
6.	5	~~3~~ 0	11	6	3
				50	29

WWW.INDIAPUSTAK.com 9560022702

P_1	P_2	P_3	P_4	P_3	P_3	P_5	P_6	P_2	P_1

0 1 2 3 4 5 6 8 11 15 21

$$\text{avg TAT} = \frac{50}{6} = 8.33$$

$$\text{avg WT} = \frac{29}{6} = 4.83$$

Ques.

P.NO.	A.T.	BT	CT	TAT	WT
1.	10	~~6~~ 0	23	13	7
2.	6	~~6~~ ~~4~~ 0	17	11	5
3.	8	~~3~~ ~~2~~ 0	11	3	0
4.	1	~~15~~ ~~12~~ 0	42	41	26
5.	9	~~2~~ 0	13	4	2
6.	4	~~8~~ ~~7~~ 0	30	26	17
				98	57

	P_4	P_6	P_2	P_3	P_3	P_5	P_2	P_1	P_6	P_4

0 1 4 6 8 9 11 13 17 23 30 42

$$\text{avg TAT} = \frac{98}{6} = 16.33$$

$$\text{avg WT} = \frac{57}{6} = 9.50$$

Ques.

P.No.	AT	BT	CT	TAT	WT
1.	12	~~2~~0	16	4	2
2.	10	~~4~~20	14	4	0
3.	8	~~7~~50	21	13	6
4.	1	~~2~~0	3	2	0
5.	5	~~11~~80	29	24	13
6.	4	~~13~~120	41	37	24
				84	45

avg TAT $= \frac{84}{6} = 14$

avg WT $= \frac{45}{6} = 7.5$

	P_4		P_6	P_5	P_3	P_2	P_2	P_1	P_3	P_5	P_6

0 1 3 4 5 8 10 12 14 16 21 29 41

Ques. Gate 2016

PNO	AT	BT	CT	TAT	WT
1	0	~~10~~7	20	20	10
2	3	~~6~~2	10	7	1
3	7	~~1~~0	8	1	0
4	8	3	13	5	2
				33	13

avg TAT $= \frac{33}{4} = 8.25$

avg WT $= \frac{13}{4} = 3.25$

what is avg TAT using SRTF algo 8.25

P_1	P_2	P_3	P_2	P_4	P_1

0 3 7 8 10 13 20

WWW.NOTESPUSTAK.com 9600127702

Ques. Gate 2007

P.NO.	AT	BT	CT	TAT	WT
1	0	~~20~~20	20	20	0
2	15	~~25~~15	55	40	15
3	30	10	40	10	0
4	45	15	70	25	10

WT of Process P_2 using SRTF 15

P_1	P_1	P_2	P_3	P_2	P_2	P_4

0 15 20 30 40 45 55 70

Ques. Consider 3 processes A, B, C to be scheduled as per SRTF algorithm. The process 'A' is known to be scheduled first, and when A has been running for '7 units' of time, the process C has arrived. The process 'C' has run for '1' unit of time, then the process B has arrived & completed running in 2 units of time. Then what could be the minimum Burst Time of process B and C?

a). 11 and 4 b). 11 and 3 c). 12 and 3 d). 12 and 4

Ans. A → 7 + 5 = 12
C → 1 + 3 = 4
B → 2

Ans: 12 and 4.

4. LONGEST JOB FIRST (L.J.F)

Criteria: Burst Time

Mode: NON preemptive.

Ques.

P.NO	AT	BT	CT	TAT	WT
1	2	6	8	6	0
2	3	4	25	22	18
3	0	2	2	2	0
4	4	8	16	12	4
5	1	5	21	20	15
				62	37

avg TAT = $\frac{62}{5}$ = 12.40

avg WT = $\frac{37}{5}$ = 7.40

P_3	P_1	P_4	P_5	P_2

0 2 8 16 21 25

Note: If the Burst Time of the processes are matching then schedule the process which has lowest Arrival time.

WWW.INDIAPUSTAK.COM 9560127702

Ques.

P.No	AT	BT	CT	TAT	WT
1	13	28	49	36	08
2.	15	17	106	91	74
3.	7	23	72	65	42
4.	12	17	89	77	60
5.	2	19	21	19	0
6.	9	13	119	110	97
				398	281

avg TAT = $\frac{398}{6}$ = 66.33

avg WT = $\frac{281}{6}$ = 46.83

	P_5	P_1	P_3	P_4	P_2	P_6

0 2 21 49 72 89 106 119

20/11/2019

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

Criteria: Burst Time

Mode: Preemptive

Ques.

PNO.	AT	BT	CT	TAT	WT
1.	1	~~2~~1	21	20	18
2.	2	~~4~~3	20	18	14
3.	3	~~6~~5	17	14	8
4.	4	~~8~~7	12	8	0
				60	40

avg TAT = $\frac{60}{4}$ = 15

avg WT = $\frac{40}{4}$ = 10

✗ wrong.

⇓

	P_1	P_2	P_3	P_4	P_4	P_3	P_2	P_1

0 1 2 3 4 5 12 17 20 21

PNO	AT	BT	CT	TAT	WT
1.	1	~~2~~1	18	17	15
2.	2	~~4~~3 ~~2~~	19	17	13
3.	3	~~6~~5 ~~4~~ ~~2~~	20	17	11
4.	4	~~8~~7 ~~6~~ ~~4~~ ~~2~~	21	17	9
				4*17	48

avg TAT = 17

avg WT = $\frac{48}{4}$ = 12.

	P_1	P_2	P_3	P_4	P_4	P_3	P_4	P_3	P_4	P_2	P_3	P_4	P_2	P_3	P_4	P_1	P_2	P_3	P_4

0 1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

✓ Right.

Ques.

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

avg TAT = $\frac{39}{3}$ = 13.

avg WT = $\frac{25}{3}$ = 8.33

What is the avg TAT using LRTF ____ ?

P_3	P_2	P_3	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3

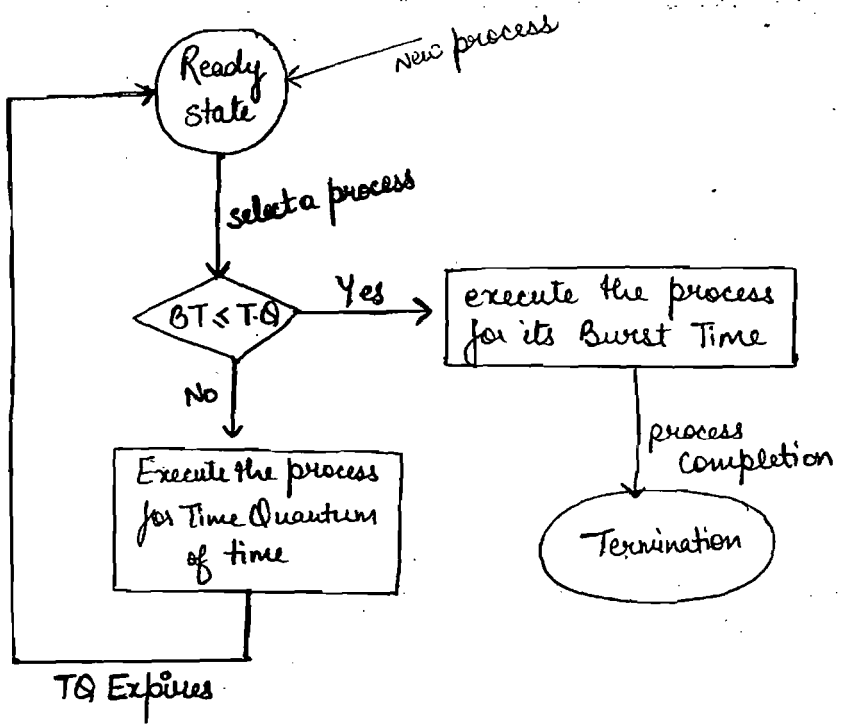
0 4 5 6 7 8 9 10 11 12 13 14

6. Round Robin Scheduling..

criteria: Time Quantum, Arrival Time (Time slice)

Mode: preEmptive.

WWW.INDIAPOSTAK.com 9560227702



Ques.

TQ = 2

P NO	AT	BT	CT	TAT	WT	RT
1	0	~~4~~ ~~2~~ ~~0~~	8	8	4	0
2	1	~~5~~ ~~3~~ ~~1~~ ~~0~~	18	17	12	1
3	2	~~2~~ ~~0~~	6	4	2	2
4	3	~~1~~ ~~0~~	9	6	5	5
5	4	~~6~~ ~~4~~ ~~2~~ ~~0~~	21	17	11	5
6	6	~~3~~ ~~1~~ ~~0~~	19	13	10	7
				65	44	20

avg TAT $= \frac{65}{6} = 10.83$

avg WT $= \frac{44}{6} = 7.33$

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

WWW.INDIAPUSTAK.com 9560127702

Gantt Chart

CS	CS	CS	CS								
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 21

Ready Queue

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

Ques.

TQ = 3

P.NO	AT	BT	TAT	WT	RT	CT
1	5	~~8~~ ~~2~~	27	22	10	32
2	4	~~6~~ ~~3~~ ~~0~~	23	17	5	27
3	3	~~7~~ ~~4~~ ~~1~~	30	23	3	33
4	1	~~9~~ ~~6~~ ~~3~~ ~~0~~	29	20	0	30
5	2	~~2~~ ~~0~~	4	2	2	6
6	6	~~3~~ ~~0~~	15	12	12	21
			128	96	32	

avg TAT $= \frac{128}{6} = 21.33$

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

avg RT $= \frac{32}{6} = 5.33$

Ready Queue

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

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

0 1 4 6 9 12 15 18 21 24 27 30 32 33

Ques. TQ=2

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

$$\text{avg TAT} = \frac{71}{6} = 11.83$$

$$\text{avg WT} = \frac{52}{6} = 8.66$$

$$\text{avg RT} = \frac{21}{6} = 3.5$$

Ready Queue

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

P_3	P_6	P_1	P_3	P_4	P_5	P_6	P_2	P_1	P_3	P_2

0 2 4 6 8 9 11 13 15 16 18 19

Ques. 31 (B2).

P.NO	AT	BT
1	0	5
2	1	7
3	3	4

TQ=2

Ready Queue

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

P_1	P_2	P_1	P_3	P_2	P_1	P_3

0 2 4 6 10 12 13 15

WWW.INDIAPUSTAK.com 9560127702

Note: 1. In the Round Robin, if the TQ is less then No. of context switches will increase and response time will be less.

2. If the TQ is large, then No. of context switches will decrease and response time will be more.

3. If the TQ is greater than all the BT ~~in~~ then Round Robin Similarly behaves like FCFS algorithm.

Ques. Consider 4 processes P_1, P_2, P_3, P_4. All arriving in the ready queue in the same order at time '0'. The Burst time of these processes 4, 1, 8, 1 respectively. Then what is the completion time of process P_1 assuming round robin scheduling with the TQ = 1.

Ready Queue: $P_1\ P_2\ P_3\ P_4\ P_1\ P_3\ P_1\ P_3\ P_1\ P_3\ P_3\ P_3\ P_3\ P_3\ P_3$.

P_1	P_2	P_3	P_4	P_1	P_3	P_1	P_3	P_1	P_3	P_3	P_3	P_3	P_3

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Ans

~~**Ques.** Consider n process At~~

Ques. Consider 'n' processes sharing the CPU in round robin algo. The context switching time is s units, then what must be the time quantum 'q' such that the no. of context switches are reduced, but at the same time each process is gaurantee to get its Turn/Job at the CPU after every 't' seconds of time?

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

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

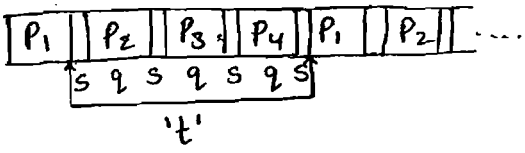
c) $q = \frac{t - ns}{n-1}$ ✓

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

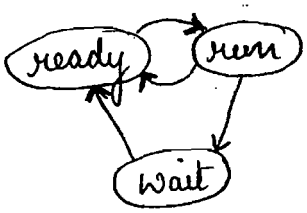
$n = 4 (P_1\ P_2\ P_3\ P_4)$

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

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



7. CPU Time, I/O Time Concept



LifeCycle

CPU Time
I/O Time
CPU Time
I/O Time
⋮
Complete

Scenario's

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

WWW.INDIANUSTAK.com 9560127702

Ques.

P.No.	AT	Execution Time		
		CPU Time	I/O Time	CPU Time
1	0	~~1~~ 0	~~2~~ 1 0	~~2~~ 1 0
2	1	~~2~~ 1 0	~~4~~ 3 2 1 0	~~5~~ 0
3	2	~~3~~ 1 0	~~6~~ 1 0	~~8~~ 0

What is completion Time of process $P_1\ P_2\ P_3$ using SRTF algo?

Note: 1). Process first spends CPU time, followed by I/O Time and followed by CPU time again.

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

P_1	P_2	P_2	P_1	P_1	P_3	P_3	P_2		P_3

0, 1, 2, 3, 4, 5, 7, 8, 13, 14, 22

P_1 — I/O: NO; P_2 — P_1; P_2P_3 — P_1; P_1P_3 — P_2; P_1P_3 — P_2; P_3 — P_2; P_2P_3; P_2 — P_3; P_3; P_3

Ans.

P.NO.	CT
1	5
2	13
3	22

Ques.

PNO	AT	Execution Time. I/O Time	CPU Time	I/O Time.
1	0	~~4~~ 40	~~14~~ 140	20
2	0	~~8~~ 80	~~28~~ 280	40
3	0	~~12~~ 120	~~42~~ 420	60

What is the completion Time of processes P_1 P_2 P_3 using SRTF also

	P_1	P_1	P_1	P_2	P_2	P_3	P_3	

0, 4, 8, 12, 18, 20, 46, 50, 88, 94

$P_1P_2P_3$; P_1 — P_2P_3; P_1P_2 — P_3; $P_1P_2P_3$; P_2P_3 — P_1; P_2P_3; P_3 — P_2; P_3; P_3

PNO	CT
1	20
2.	50
3.	94

WWW.INDIAPLUSTAK.com 9660127702

Q.28 (B1).

P.NO.	AT	I/O Time	CPU Time	I/O Time
1	0	~~2~~ 20	~~7~~ 70	~~1~~ 10
2	0	~~4~~ 40	14	2
3	0	~~6~~ 60	21	3

	P_1	P_2	P_3		

0, 2, 9, 23, 44, 47

$P_1P_2P_3$; P_1 — P_2P_3; P_2P_3 — P_1; P_3 — P_2; P_3

% CPU remain idle $= \frac{5}{47} \times 100$

$= 10.6\%$

Ques.

P.NO	AT	Execution Time		
		CPU Time	I/O Time	CPU Time.
1.	0	~~5~~ 2 0	~~5~~ 3 0	2 0
2.	3	2 0	~~22~~ ~~19~~ 17 12 10	2 0
3.	7	~~8~~ ~~5~~ 0	0	0
4.	25	~~8~~ 5	2	1

What is the completion time of processes P_1 P_2 P_3 P_4 using SRTF algo ?

P_1	P_1	P_2	P_3	P_1	P_3		P_4	P_2	P_4		P_4

0 3 5 7 10 12 17 25 29 31 36 38 39

$\uparrow P_1$ $\uparrow P_1P_2$ $\uparrow P_2$ $\uparrow P_3$ $\uparrow P_1P_3$ $\uparrow P_3$ $\uparrow$ $\uparrow P_4$ $\uparrow P_2P_4$ $\uparrow P_4$ $\uparrow P_1$

P_1 P_1P_2 P_2 P_2 P_2 P_2 P_1

PNO	CT
1	12
2	31
3	17
4	39

8. Priority Based Scheduling (NON preemptive)

criteria : priority

Mode: NON preemptive

WWW.INDIANPUSTAK.com 9560127788

Ques.

priority	P.NO	AT	BT	CT	TAT	WT
3	1	0	2	2	2	0
1	2	1	4	25	24	20
6	3	2	6	8	6	0
5	4	3	8	21	18	10
Highest ← 8	5	4	5	13	9	4
					59	34

avg TAT = $\frac{59}{5} = 11.80$

avg WT = $\frac{34}{5} = 6.80$

P_1	P_3	P_5	P_4	P_2

0 2 8 13 21 25

Note: If the priority of the processes are matching, then schedule the process which has lowest Arrival Time.

Ques.

priority	PNO	AT	BT	CT	TAT	WT
4	1	4	6	19	15	9
7 (High)	2	6	3	9	3	0
6	3	3	4	13	10	6
6	4	2	2	6	4	2
1	5	1	3	4	3	0
3	6	2	2	21	19	17
					54	34

$$\text{avg TAT} = \frac{54}{6} = 9.00$$

$$\text{avg WT} = \frac{34}{6} = 5.66$$

	P_5	P_4	P_2	P_3	P_1	P_6

0 1 4 6 9 13 19 21

9. Priority Based Scheduling (preEmptive)

Criteria : priority

Mode : preEmptive.

Ques.

priority	P.NO	AT	BT	CT	TAT	WT
2	1	1	8~~4~~	25	24	19
3	2	2	6	21	19	13
4	3	2	3~~2~~	15	13	10
6	4	3	4~~3~~	11	8	4
5	5	3	2	13	10	8
8 (high)	6	4	4	8	4	0
					78	54

$$\text{avg TAT} = \frac{78}{6} = 13$$

$$\text{avg WT} = \frac{54}{6} = 9$$

	P_1	P_3	P_4	P_6	P_4	P_5	P_3	P_2	P_1

0 1 2 3 4 8 11 13 15 21 25

Ques.

Priority	PNO.	AT	BT	CT	TAT	WT
2	1.	1	9	50	49	40
3	2.	2	8~~7~~	37	35	27
5	3.	3	7~~6~~	30	27	20
2	4.	0	6~~4~~	41	41	35
6	5.	4	8~~7~~	24	20	12
7 (High)	6.	5	12	17	12	0
					184	134

$$\text{avg TAT} = \frac{184}{6} = 30.66$$

$$\text{avg WT} = \frac{134}{6} = 22.33$$

P_4	P_4	P_2	P_3	P_5	P_6	P_5	P_3	P_2	P_4	P_1

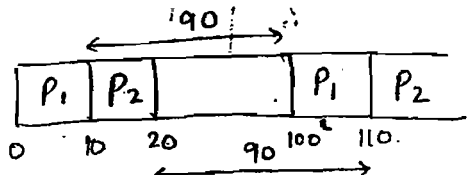
0 1 2 3 4 5 17 24 30 37 41 50

WWW.INDIAPUSTAK.COM 9560187704

Q.29 (81)

	AT	CPU	I/O	P
P_1	0	10	90	10
P_2	0	10	90	9

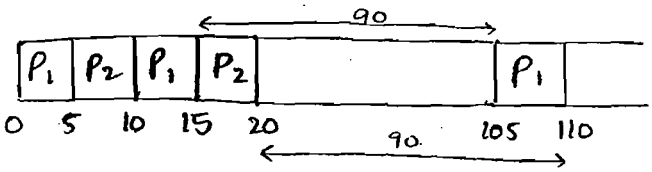
FCFS:



Idle Time $= \frac{80}{100} = 0.8$

↳ Static Priority
↳ SRTF

RR:



Idle Time $= \frac{85}{100}$
$= 0.85$

10. Highest Response Ratio Next (H.R.R.N)

Criteria: Response Ratio

Mode: Non preemptive.

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

W = waiting time
S = Service time and Burst Time.

WWW.INDIAPLUSTALK.com 8560127702

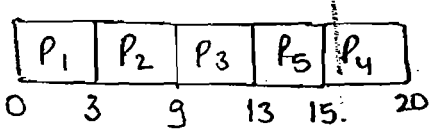
Note: The above algorithm favours shorter jobs & limits the waiting time of longer jobs.

Ques.

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

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

avg WT $= \frac{20}{5} = 4$



$R.R_3 = \frac{W+S}{S} = \frac{5+4}{4} = \frac{9}{4} = 2.25$

$R.R_4 = \frac{3+5}{5} = \frac{8}{5} = 1.6$

$R.R_5 = \frac{W+S}{S} = \frac{1+2}{2} = 1.5$

$R.R_4 = \frac{7+5}{5} = \frac{12}{5} = 2.4$

$R.R_5 = \frac{5+2}{2} = \frac{7}{2} = 3.5$

Ques.

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

$$\text{avg TAT} = \frac{33}{5} = 6.6$$

$$\text{avg WT} = \frac{16}{5} = 3.2$$

	P_1		P_2	P_5	P_3	P_4

0 1 3 4 10 11 14 19

$R.P_3 = \frac{5+3}{3} = \frac{8}{3} = 2.6$

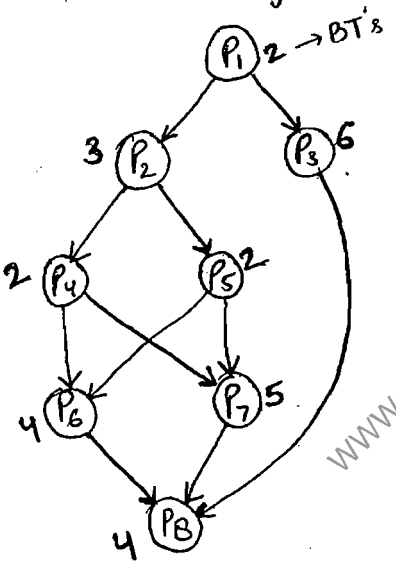
$R.P_4 = \frac{4+5}{5} = \frac{9}{5} = 1.8$

$R.P_5 = \frac{2+1}{1} = 3$

$R.P_3 = \frac{6+3}{3} = 3$

$R.P_4 = \frac{5+5}{5} = 2$

Ques. Consider the following dependency graph between the processes. At what time all the processes complete their execution by using 2 CPU processor's ?



Note. 1). one process can not use 2 CPU's at the same time.

2). use NON-preEmptive mode.

a). 17 b). 18 c). 19

d). 20 e). 21

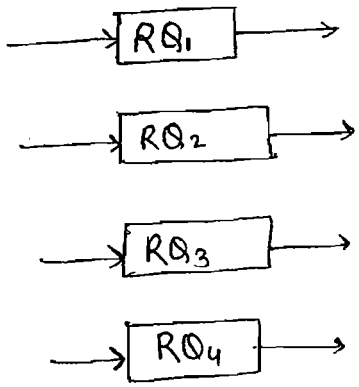
CPU1	P_1	P_2	P_4	P_5	P_5	P_6		P_8
CPU2		P_3	P_3	P_3		P_7	P_7	

0 2 5 7 8 9 13 14 18

WWW.INDIAPUSTAK.COM 9560427102

21/11/2019

Multi-level Queue Scheduling



- Depending on the priority of the process, in which particular ready queue process has to be placed. will be decided.
- The high priority processes will be placed in the top level ready queue and low priority process will be placed in the bottom level ready queue.
- Only after completion of all the processes from the 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 the bottom level ready queue will suffer from Starvation.

WWW.INFOPUSTAK.com 9560127702

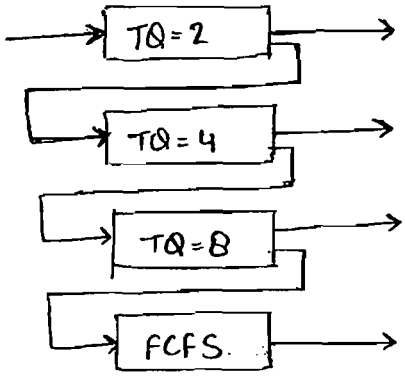
Starvation: The indefinite waiting of a process is called as Starvation.

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

Ageing: If the waiting time of a process increases, then the priority of the process increase.

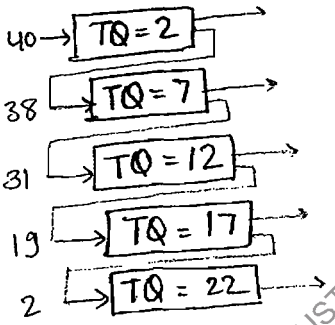
If the age of the process increases, then by increasing the priority, process will definately get chance to execute and starvation problem will be avoided.

Multi-level Feedback Queue Scheduling :



Note: In the above algorithm every process will definately get a chance to execute for some amount of time. But still there is a possibility of starvation.

Q.8 (79)



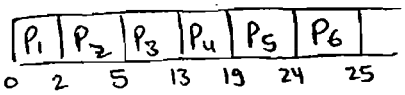
Ans. 4, 5

WWW.INDIAPUSTAK.com 9560127702

Ques.

P.NO.	AT	BT	CT
1	0	2	2
2	1	3	5
3	2	8	13
4	3	6	19
5	4	5	24
6	5	1	25

what is the throughput of the system using FCFS ___24.___



$$\boxed{\text{Throughput} = \frac{\#\text{ of processes}}{\max(CT) - \min(AT)}}$$

$$\text{Throughput} = \frac{6}{25-0} = \frac{6}{25} = 0.24.$$

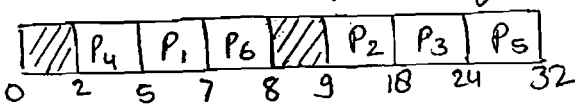
$$\boxed{\% = 24\%}$$

$$\boxed{\text{Schedule Length} = \max(CT) - \min(AT)}$$

Ques.

PNO.	AT	BT
1	3	2
2	9	9
3	12	6
4	2	3
5	15	8
6	3	1

What is the throughput of the system using FCFS ?



$$\text{Throughput} = \frac{6}{32-2} = \frac{6}{30} = 0.20$$

$$\boxed{\% = 20\%}$$

Imp:

Algorithm	Starvation
1) FCFS	No
2). NP → SJF	Yes
3). SRTF	Yes
4). RR	No
5). NP → LJF	Yes
6). LRTF	No
7). NP → Priority	Yes.
8). pre-priority	Yes.
9). HRRN	No
10). MLQ	Yes.
11). MLFBQ.	Yes.

WWW.INDIAPUSTAK.com 9560127702

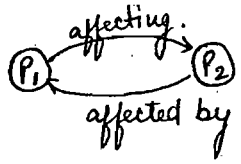
Note: Majority of the operating systems practically implement Round Robin Scheduling. eg: Windows, UNIX, Linux etc.

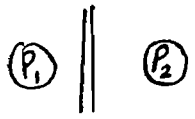
Synchronization :

- Processes are categorized into 2 types.

Processes
- Co-operative
- Independent

- The execution of 1 process affects or affected by other process then those processes are said to be Co-operative processes. otherwise they are said to be independent process.





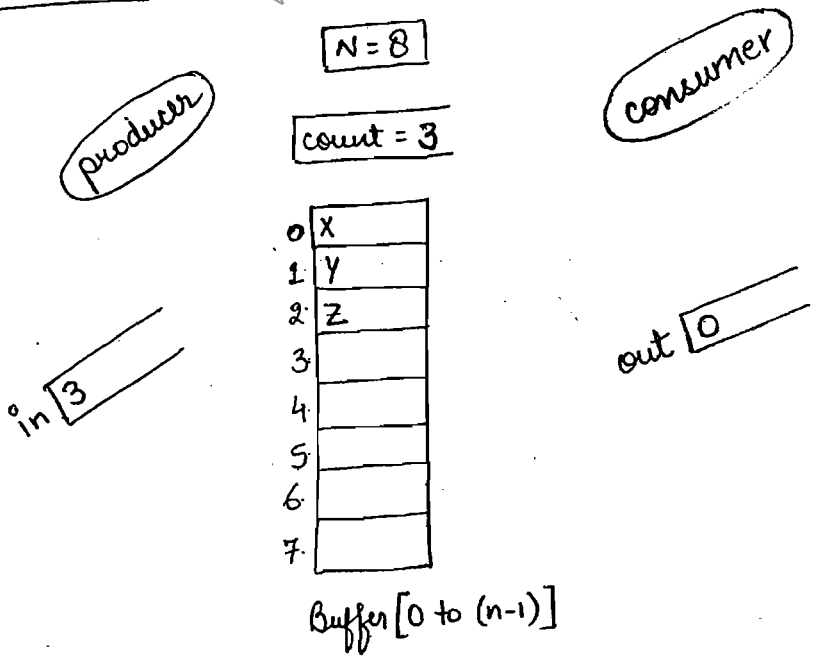
- Understanding Synchronization :

1). The problems arises not having synchronization between the processes.
2). The conditions to be followed to achieve synchronization.
3). The solutions (wrong or right).

WWW.INDIAPUSTAK.com 9560127799

(1). The problems arises not having synchronizat^n b/w the processes.

The Producer - Consumer Problem :



```
int count = 0;
void producer (void).
{
    int itemp;
    while (true)
    {
        produce-item (itemp);
        while (count = = N) ;      → looping until false
        Buffer [in] = itemp;
        in = (in+1) Mod N;
        count = count + 1;
    }
}
```

I. Load R_p, M[count]
II. INCR R_p
III. Store m[count], R_p

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

I. Load R_c, M[count]
II. DECR R_c
III. Store M[count], R_c

WWW.UNIAPUNK.com 9560127702

- **In** is a variable used by producer to identify Not empty seat in the buffer.
- **Out** is the variable used by consumer to identify from where it has to consume the item.
- **count** is a variable used by both producer & consumer to identify No. of items present in the buffer at any point of time.

Shared Resources

1. Count Variable
2. Buffer.

Two Conditions to be followed :

1). If the Buffer is full, producer is Not allowed to produce the item in the buffer.

2). If the Buffer is empty, then consumer is Not allowed to consume the item from the buffer.

Analysis

Universal Assumption:

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

1). $\boxed{\text{Count} = \cancel{3}\,\cancel{2}\,4}$

itemp = A

item C = X

P → I

P → II

C → I

C → II

C → III

P → III

Rp $\boxed{\cancel{3}\,4}$

Rc $\boxed{\cancel{3}\,2}$

in $\boxed{\cancel{3}\,4}$

out $\boxed{\cancel{0}\,1}$

0	
1	Y
2	Z
3	A
4	
5	
6	
7	

Buffer

1. **Inconsistency Problem:**

The producer & consumer are Not properly synchronized while sharing common variable count. Hence, it is leading to inconsistency.

WWW.INDIAPUSTAK.com 9560127703

Printer Spooler Daemon :

Daemon : Background Process

$\boxed{N=8}$

processes $P_1 P_2 P_3 P_4 \ldots$

printer

in $\boxed{3}$

out $\boxed{0}$

0	a.doc
1	b.doc
2	c.doc
3	
4	
5	
6	
7	

Spooler Directory [SD]

Enter file

I. Load R_i , m[in]

II. Store SD[R_i] , "F-N"

III. INCR R_i

IV. Store m[in] , R_i

i : Respective Process Register

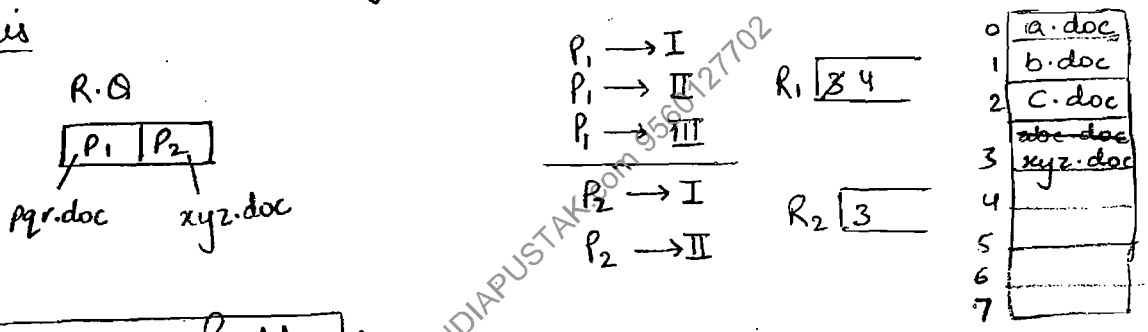
F-N : File Name

- $\boxed{IN}$ is a variable used by all the processes to identify Next empty slot in the spooler Directory.
- $\boxed{out}$ is a variable used by the printer to identify from where it has to print the document.

Shared Resources

1. IN Variable
2. Spooler directory.

Analysis



WWW.INDIAPUSTAK.COM 9560127702

2. Loss of Data Problem :

The processes are not properly synchronized while sharing the common variable IN. Hence, it is leading to the problem of loss of Data.

3. Deadlock Problem :

If the processes are not properly synchronized while sharing common resources or common variables, then it is also possible for deadlock.

Definations

1). Critical Section : The portion of the program text where shared resources or shared variables will be placed.

Eg : count = count + 1 ;

2). Non-critical section : The portion of program text where ~~independe~~ independent code of the process will be placed.

Eg : in = (in+1) mod N.

3). Race Condition : The final value of any variable depends on execution sequence of the processes. This type of condition is called as RACE CONDITION.

Eg :

Count = 4	Count = 2
P – I	C – I
P – II	C – II
C – I	P – I
C – II	P – II
C – III	P – III
P – III	C – III.

Note : To avoid the problem of RACE Condition, only 1 process should be present inside the critical section at any point of time.

(2). Condition to be followed to achieve Synchronization.

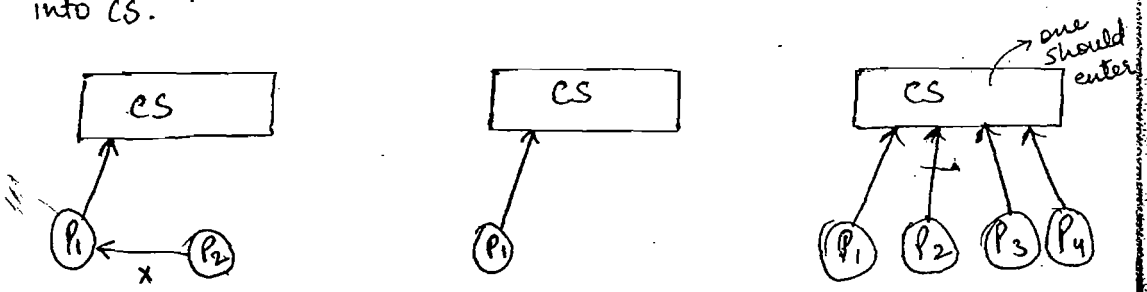
WWW.INDIANOTES.TK 9560127002

1. Mutual Exclusion (ME):

- No two processes may be simultaneously present inside the critical section at any point of time.
- Only one process should be present inside the critical section at any point of time.

2. Progress :

- No process running outside the critical section should block other interested process from entering into C.S when CS is free.
- If there is only one process trying trying to enter into CS then it should be definately allowed to enter into CS.
- If 2 or more processes are trying to enter into CS then one process should be definately allowed to enter into CS.



3. **Bounded Waiting** :

- No process should have to wait forever to enter into Critical section
- There should be a bound on ~~giving~~ getting chance to enter into CS.
- Some process is indefinately waiting to enter into CS, because CS is always busy by some other processes. This situation should not arrise.
- If the bounded waiting is not satisfied then it is possible for starvation.

4. No assumption related to Hardware and processor speed.

3. Solutions :

I. Software Type
   a) Lock Variables
   b) Strict Alteration or Decker's algorithm.
   c) Petersons Algorithm

II. Hardware Type
   a) TSL Instruction Set
      → Test and Set Lock.

III. OS Type
   a) Counting Semaphore.
   b) Binary Semaphore.

IV. Programming Language Compiler Support Type
   a). Monitors.

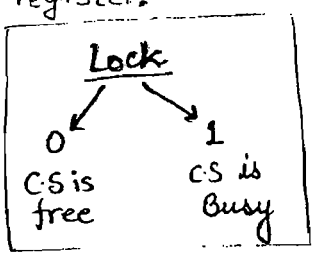
WWW.INDIAPUSTAK.com 9560127702

I. Software Type Solutions :

a). Lock Variables :

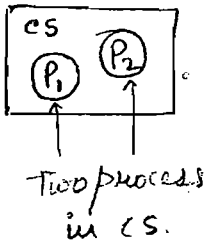
Entry Section

I. Load Ri, M[lock] → Respective process register.
II. CMP Ri, #0
III. JNZ to step 1
IV. Store m[lock], #1
V. c.s
VI. Store m[lock], #0



Analysis

Lock = ~~0~~, ~~1~~, 1



$P_1 \rightarrow$ I
$P_1 \rightarrow$ II
$P_2 \rightarrow$ III

$P_2 \rightarrow$ I
$P_2 \rightarrow$ II
$P_2 \rightarrow$ III
$P_2 \rightarrow$ IV
$P_2 \rightarrow$ V

$P_1 \rightarrow$ IV
$P_1 \rightarrow$ V

R_1 [0]

R_2 [0]

ME is Not satisfied.

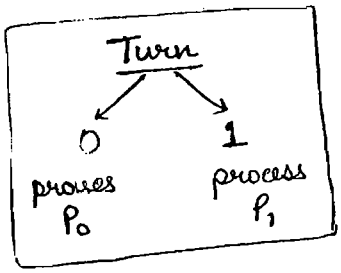
We have proved that both the processes are coming into CS at the same time. Hence, mutual Exclusion is not satisfied and solution is considered to be incorrect.

b). Strict Alteration or Deaker's Algorithm:

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

Process 'P_0' Code	Process 'P_1' code

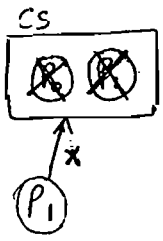
while(True)
{
Non_CS();
while (turn != 0);
CS
turn = 1;
}
	while(True)
	{
	Non_CS();
	while (turn != 1);
	CS
	turn = 0;
	}



WWW.INDIAPGSTAK.com 9560127702

Analysis

turn = ~~0~~, ~~1~~, 0



1. ME is satisfied
2. Progress is Not satisfied.

We have proved that progress is Not satisfied and sol^n is considered to be incorrect.

Important Points:

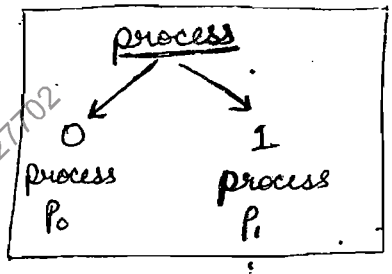
1. The pre-Emption is just a temporary stop. The process will come back and continue remaining execution.
2. If there is any possibility of solution becoming wrong by taking any preEmption then consider the preEmption.
3. If any solution is having deadlock, then progress is not satisfied.

22/11/19

C). Peterson's Algorithm (2 process Solution)

Initially

interested [0] = false;
interested [1] = false;




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

- TURN is the shared variable used by both the processes P_0 and P_1

WWW.INDIAPUSTAK.com 9560127702

Analysis ①:-

interested [0] = ~~False~~ True;
interested [1] = ~~False~~ True.

turn = ~~0~~ 1.

process P_0	Process P_1
other = 1	other = 0

CS
P_0

Analysis ②:

interested [0] = ~~False~~ True
interested [1] = ~~False~~ True.

turn = ~~1~~ 0

process 0	process 1
other = 1	other = 0

CS
P_1

1. ME is satisfied
2. Progress
3. Bounded Waiting is satisfied

WWW.INDIANUSTPK.COM 9560127702

- We have proved that all the three conditions are satisfied in the above solution & it is properly providing synchronization to the processes to access the common critical section.

Ques. Assume that both the processes P_0 and P_1 are trying to enter into critial section at the same time. Then which process will enter into critical section first.

a). The process which executes stmt 2 first.
b) The process which executes stmt 3 first
c) The process which executes stmt. 4 first.
d). We cannot say.

interest [0] = ~~F~~ T
interest [1] = ~~F~~ T

turn = ~~1~~ 0

Process 0	Process 1
other = 1	other = 0

P_0 P_1
↓ ↓

1. int other
2. other = 1 − process;
3. interested [process] = TRUE; < P_0, P_1
4. turn = process < P_1, P_0
5. while (turn == process && interest [other] == True);

P_0: T T
P_1: F T

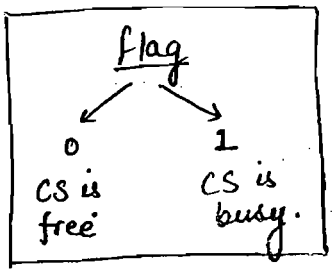
II. Hardware Type Solutions

a). TSL Instruction Set → Test & Set Lock.

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

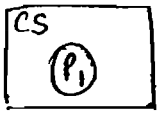
Entry Section :

I. TSL R_i , M[Flag].
II. CMP R_i , #0
III. JNZ to step ①
IV. $\boxed{\text{CS.}}$
V. Store M[flag] ; #0



Analysis.

flag = ∅ ~~1~~, 1



$P_1 \to$ I
$P_1 \to$ II
$P_1 \to$ III
$P_1 \to$ IV

R_1 [0]

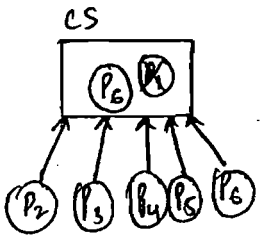
$P_2 \to$ I
$P_2 \to$ II
$P_2 \to$ III.

R_2 [1]

1. ME is satisfied.
2. Progress is satisfied.
3. Bounded Waiting is Not satisfied.

Bounded Waiting

Flag = ∅ ~~1~~ 1



$P_1 \to$ I
$P_1 \to$ II
$P_1 \to$ III
$P_1 \to$ IV.

R_1 [0]

$P_2 \to$ I
$P_2 \to$ II
$P_2 \to$ III

R_2 [1]

1. ME is satisfied
2. Progress is satisfied
3. Bounded Waiting is Not satisfied.

Note : 1. If some process is in the critical section then all the other processes which are ~~tr~~ trying to enter into CS will be repeatedly checking for I, II, III inst^n.

WWW.INDIAPOSTAK.com 9560127702

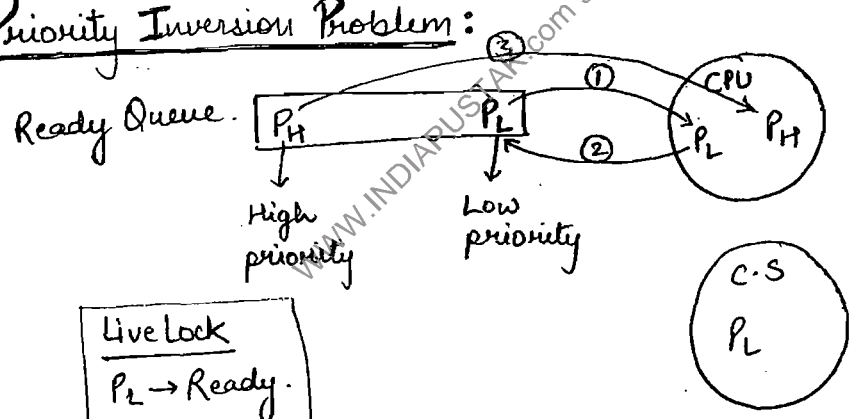
2. These processes are busy in checking those instructions & waiting to enter into C.S. This is called as "Busy Waiting"

3. Busy Waiting is also called as "Spin Lock".
Because of Busy waiting, CPU cycles will be wasted.

4. To avoid the problem of Busy Waiting, the concept of "Semaphore" will be used.

Solution	ME	Progress	B.W
Lock Variables.	X	✓	X
Strict Alteration (or) Decker's Algo.	✓	X	✓
Peterson's Algorithm	✓	✓	✓
TSL Instruct^n Set.	✓	✓	X

Priority Inversion Problem:



Live Lock
$P_L \rightarrow$ Ready.
$P_H \rightarrow$ Run.

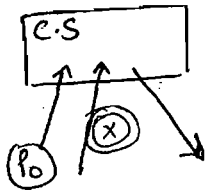
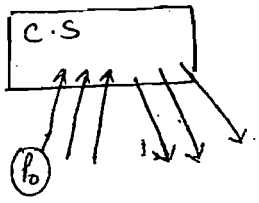
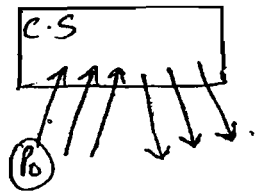
Solution

Priority Inheritance

$P_H \rightarrow 20$

$P_L \rightarrow$ ~~2~~ 20. (Inheriting the priority from P_H).

WWW.INDIAPUSTAK.com 9560127702

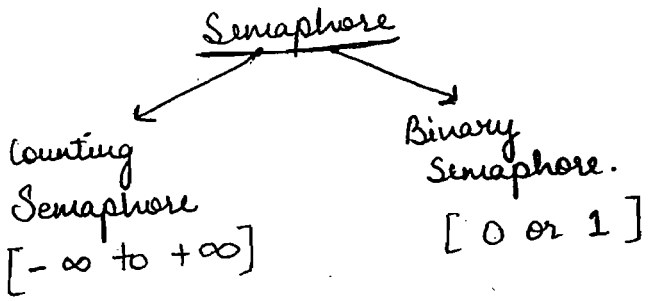
Strict alteration (or) Decker's algo	Peterson's Algo.	TSL Instn set
Process ⇒ P_0, P_1	Processes ⇒ P_0, P_1	Process ⇒ P_0, P_1, P_2 .....
Progress ✗	Progress ✓	Progress. ✓
		

<u>Note</u>: We can apply above techniques only after checking for deadlock.

III. OS Type Solutions:

- Semaphore is an integer variable used by various processes in a mutual exclusive manner to achieve synchronization.
- Improper usage of semaphore will also give wrong results.
- Semaphore is categorized into two types.
  1. Counting Semaphore
  2. Binary Semaphore

WWW.INDIANPUSTAK.COM 9969127702



- The two different operations performed over semaphore variable:

  1). Down(); or Wait(); or P();

  2). up(); or Signal(); or V(); or Release();

a). Counting Semaphore :

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

```
Up(Semaphore s).
{
    s.value = s.value + 1;
    if (s.value ≤ 0)
    {
        Select the process
        from Suspended List
        and Wakeup();
    }
}
```

- After performing down operation if the process is Not getting suspended then it is called as successful down operation.
- If it is successful down operation then ossly process will continue further execution in the code.
- After performing down operation, if the process is getting suspended then it is called as unsuccessful Down operation.
- If it is unsuccessful Down operation, then process will Not continue further execution in the code.
- Positive value of the Semaphore indicates No. of successful down operations we can perform.
- There is No unsuccessful up operations and up operation is always successful.
- The process performing up operation will definately continue further execution.

WWW.INDIABUSTAR.COM 9581212702

Ques. Consider a system which has a counting semaphore S = +17. The various semaphore operations like

23P, 14V, 6P, 9V, 3P, 7V are performed then what is the final value of the counting semaphore.

$$S = 17 - 23 + 14 - 6 + 9 - 3 + 7$$

$$S = 15$$

Note : The negative value of counting semaphore indicates the No. of processes blocked.

Eg: $S = \cancel{1} \cancel{0} \cancel{1} -2$

P_1	P_2	P_3
P	P	P
✓	✗	✗

S.L → P_2 P_3.

Ques. Consider the counting semaphore variable 'S'. The various Semaphore operations like 12 V (signal), 20 P are performed. Then, what is the largest initial value of the semphore S. So, that 1 process will be blocked _____ ?

$S + 12 - 20 = -1$

$S = +7$

b). **Binary Semaphore:**

```
Down (Semaphore s)
{
  if (s.value == 1)
      s.value = 0;
  else
    {
      Block the process and
      place it's P.C.B in
      the suspended list;
    }
}
```

```
up ( Semaphore s)
{
  if (Suspended list is empty)
      s.value = 1;
  else
    {
      select a process from
      suspended list and
      wakeup();
    }
}
```

WWW.INDIAPUSTAK.com 9501277702

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

~~There is~~

- There is NO Unsuccessful up Operation and up Operation is always successful.
- The process performing up operation will definately continue further execution.

Eg: S = ~~1~~ ~~0~~ ~~1~~ 1

P_1	P_2	P_3	P_4	P_5	P_6	P_7
P	P	P	V	V	V	V
✓	X	X	✓	✓	✓	✓

SL: P_3 P_5
✓ ✓

Important Points: (Applicable for both counting & Binary Semaphore).

1. Every Semaphore variable will have its own suspended list.
2. The down and up operations are atomic.
3. Every time when they perform one up operation, 1 process will wakeup from the suspended list and this will be based on first in first out (FIFO)
4. If two or more processes are in the suspended list and there is NO other process to wakeup these processes, then they are said to be involved in the deadlock.
5. When the process is in the suspended list, then it will be placed in the wait state & when the process is wakeup then it will be placed in the ready state.

www.INDIAPUSTAK.com 9566027702

Ques. Each Process P_i $i=1$ to 9 executes the following code.

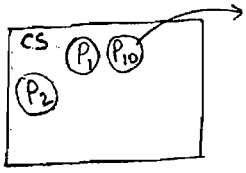
```
while(true)
{
  P(mutex);
  [CS]
  V(mutex);
}
```

The process P_{10} executes the following code

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

Analysis

mutex = ~~1~~ ~~0~~ ~~1~~ 0



P_{10} will go and come. and calls everyone.

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

Note: Initial value of binary semaphore mutex = 1;

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

Ques. if P_{10} code change a bit, then what will be the answer of the above question.

```
while (True)
{
  V(mutex);
  [ C.S. ]
  P(mutex);
}
```

Analysis
mutex = ~~1~~ ~~0~~ ~~1~~ 0

CS: P_1 P_{10} P_2

Ans: (b). 3.

Ques. if P_{10} code change a bit, then what will be the answer of the above question.

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

Analysis
mutex = ~~1~~ 0

CS: P_1

Ans: (e) 1.

WWW.INDIAPUSTAK.com 9560027702

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

Process 'P' code	Process 'Q' code

while (True)
{
w: _____
print ('0');
print ('0');
x: _____
}
	while (True)
	{
	y: _____
	print ('1');
	print ('1');
	z: _____
	}

What should be the binary semaphore operations on W, X, Y, Z respectively. and what must be the initial values of binary semaphore S and T inorder to get print output always as 0011001100.....

a). W = P(S), X = V(S), Y = P(T), Z = V(T), S = T = 1;
b). W = P(S), X = V(T), Y = P(T), Z = V(S), S = T = 1;
c). W = P(T), X = V(T), Y = P(S), Z = V(S), S = 1, T = 0;
✓ d). W = P(T), X = V(S), Y = P(S), Z = V(T), S = 0, T = 1;

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

<u>Ques</u>. Cosider two concurrent processes P and Q executing their respective codes.

process P code	process Q code

while (true)
{
W: ____
print('0')
print('0')
X: ____
}
	while (True)
	{
	Y: ____
	print (1)
	print (1)
	Z: ____
	}

WWW.INDIAPUSTAK.COM 9560127702

Which of the following will ensure that, the output string never contains the substring of the form 01^n0 or 10^n1 where n is odd?

a). W = P(S), X : V(S), Y = P(T), Z = V(T); S = T = 1;
b). W = P(T), X : V(S) Y = P(S), Z = V(T); S = T = 1;
✓ c). W = P(S), X : V(S) Y = P(S), Z = V(S); S = 1;
d). W = V(S), X : P(T) Y: V(T), Z = P(S); S = 1, T = 1;

W: P(S) Y: P(S) S = 1
X: V(S) Z: V(S)

23/11/2019

Q.16 (90) **Case I** $S_0 = \cancel{1}\cancel{0}\cancel{1}\cancel{0}_0$ $S_1 = \cancel{0}\cancel{1}\cancel{0}1$ $S_2 = \cancel{0}\cancel{1}\cancel{1}\cancel{0}1$

P_0	P_1	P_2
anytime	Once ⊗	Once ⊗

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

o/p: 0 0 0

Case II $S_0 = \cancel{1}\cancel{0}\cancel{1}\cancel{1}0$ $S_1 = \cancel{0}\cancel{1}\cancel{0}1$ $S_2 = \cancel{0}\cancel{1}\cancel{0}1$

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

o/p: 0 0

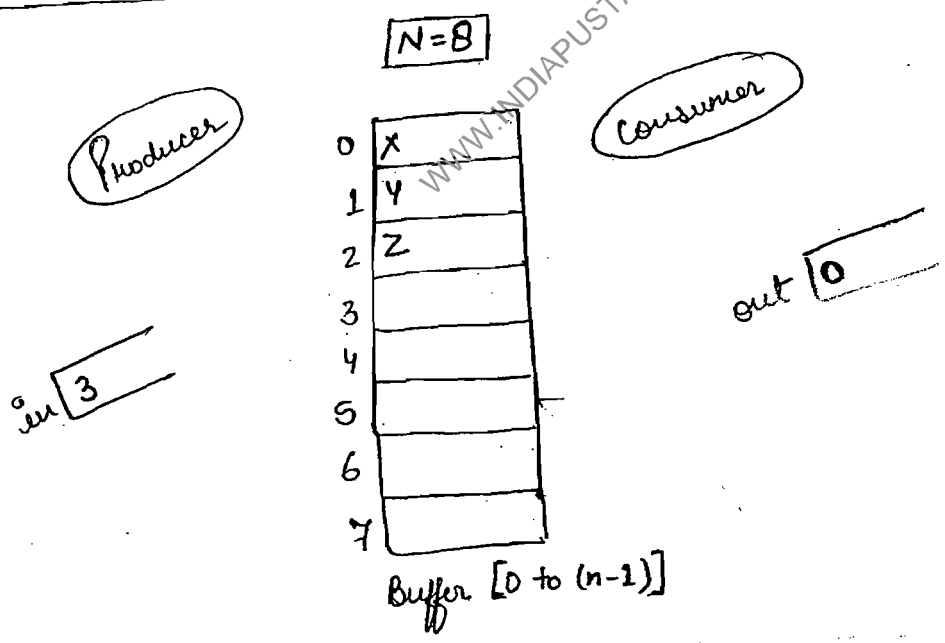
Ans: (a) at least twice.

⊗ at least once cannot be the right because zero will print at least twice in every condition.

Classical Process of IPC [Interprocess Communication]

Producer-Consumer with Semaphore :

WWW.INDIAPUSTAK.com 9560127702



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

```
Semaphore mutex = 1;
Semaphore Empty = N;
Semaphore Full = 0;
void producer(void)
{
    int itemp;
    while (true)
    {
        produce_item(itemp);
        down(Empty);
preempt ---> down(mutex);
        Buffer[in] = itemp;
        in = (in+1) mod N;
        up(mutex);
        up(full);
    }
}
void consumer(void)
{
    int item c;
    while (true)
    {
        down(Full);
        down(Mutex);
        Item c = Buffer[out];
        out = (out+1) mod N;
        up(Mutex);
        up(Empty);
        process_item(itemc);
    }
}
```

WWW.INDIAPUSTAK.com 9560127702

Analysis:

mutex = ~~2~~ ~~0~~ ~~1~~ ~~0~~ ~~0~~ 1

Empty = ~~3~~ ~~4~~ ~~5~~ 5

full = ~~3~~ ~~4~~ ~~3~~ ~~2~~ 3

8 8 8 8

Itemp = ~~A~~, B

Itemc = ~~X~~, Y

in = ~~3~~ ~~4~~ 5

out = ~~0~~ ~~1~~ 2

0	~~Z~~
1	~~Z~~
2	Z
3	A
4	B
5	
6	
7	

Ques. What happens if we interchange down(empty), down(mutex) in the producer code?

a). No problem, the solution still works correct.
b). Both producer & consumer will access the buffer at the same time.
c). some of the items produced by the producer will be lost.
d). It is possible for deadlock.

if Buffer is full
mutex = 0
full = 8
Empty = 0-1.

In this both are in Suspended List.

P C

Ques. what happens if we interchange up(mutex), up(full) in the producer code?

Ans. a). No problem, the solution still works correct.

Ques. Interchange of down(full), down(mutex) in consumer code.

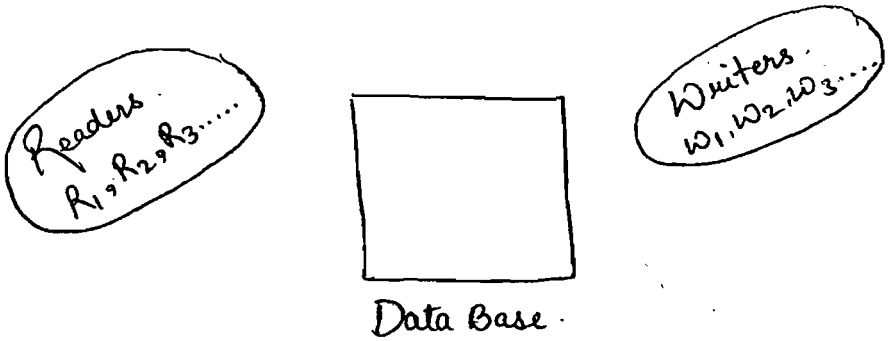
Ans. d) deadlock

Ques. Interchange of up(mutex), up(Empty) in the consumer code.

Ans. a). No problem.

WWW.INDIAPUSTAK.com 9560127702

Reader's-Writers:



- rc is integer variable represents Reader's count.
- mutex is a binary semaphore used by the readers in the mutual exclusive manner.
- db is binary semaphore used by the readers and writers in a mutual exclusive manner.

```
int rc = 0;
Semaphore mutex = 1;
Semaphore db = 1;
void reader (void)
{
  while (true)
   {
    down (mutex);
    rc = rc + 1;
    if (rc == 1) down (db);
    up (mutex);

    [ D.B ]

    down (mutex); <-
    rc = rc - 1;  <-
    if (rc == 0) up (db);
    up (mutex);
   }
  }
```

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

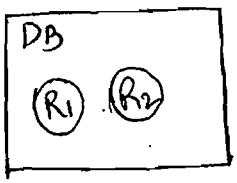
WWW.INDIAPUSTAK.com 9560127702

4 conditions to be followed:

1). R — W ✗
2). R — R ✓
3). W — R ✗
4). W — W ✗

Analysis (1).

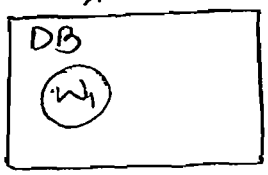
rc = ~~0~~ ~~1~~ 2
mutex = ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1
db = ~~1~~ 0



(1) R → W ✗
(2) R → R ✓

Analysis (2)

rc = ~~0~~ 1
mutex = ~~1~~ 0
db = ~~1~~ 0



(3) W → R ✗
(4) W → W ✗

Ques. What happens if we interchange down(mutex), rc = rc + 1 in the readers code.

a). No problem, Sol^n ~~still~~ still works correct.

b). Multiple readers are not allowed into DB

c). Both reader & writer will enter into DB at the same time.

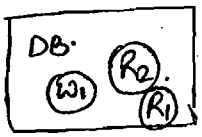
d). It is possible for deadlock

R_1 R_2 W_1

rc = ~~0~~ ~~1~~ 2

mutex = ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

db = ~~1~~ 0.



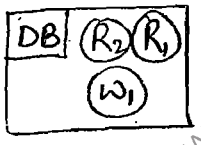
Ques. What happens if we interchange if condition and up(mutex) in the reader's code.

Ans. (c) Both reader & writer will enter into DB

rc = ~~0~~ ~~1~~ 2

mutex = ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

db = ~~1~~ 0



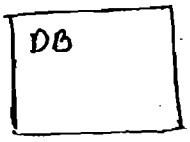
Ques. What happens if we interchange down(mutex) and rc = rc − 1; in the reader's code.

Ans. (d). Deadlock.

rc = ~~0~~ ~~1~~ ~~0~~ 1

mutex = ~~1~~ ~~0~~ ~~1~~ 0

db = ~~1~~ 0.



- reader(2) deadlock in if condition.
- reader(1) deadlock in second down(mutex)
- writer(1) deadlock in down(db);

Ques. Consider the following code used by classical Readers & Writers Which of the following is True?

a). No. problem, code is working correct

b) Multiple writers are allowed into DB at the same time

c) Both reader & writer will enter into DB at the same time.

d). It is possible for deadlock.

WWW.INDIAPUSTAK.com 9560127700

```
int rc=0;
int wc=0;
semaphore mutex = 1;
semaphore r = 1;
semaphore db = 1;

void reader(void).
{
  while (true)
    {
      down(r);
      down(mutex);
      up(r);
      rc= rc+1;
      if (rc==1) down(db);
      if (wc==1)
        {
          down(r);
          up(db);
        }
      up(mutex);
      [D.B]
      down(mutex);
      rc = rc-1;
      if (rc==0) up(db);
      up(mutex);
    }
}
```

```
~~void~~
void writer (void).
{
  while (True)
    {
      down(r);
      wc = wc+1;
      down(db);
      [DB]
      up(db);
      wc=wc-1;
      up(r);
    }
}
```

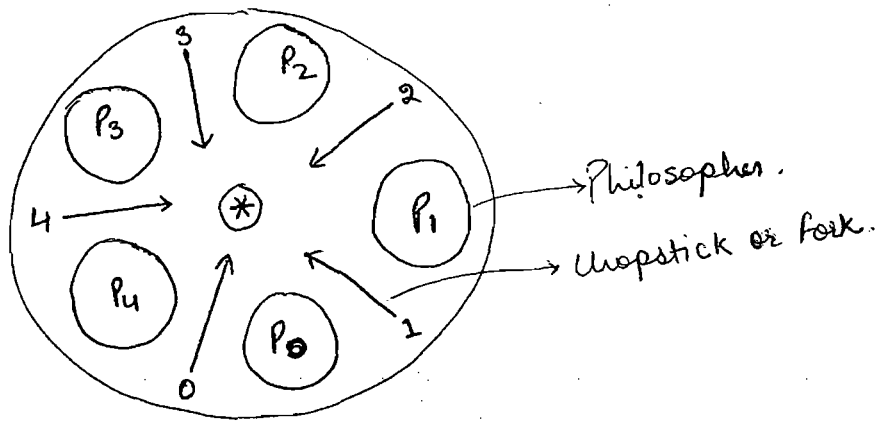
Analysis

rc = ~~0~~ 1
wc = ~~0~~ 1
mutex = ~~1~~ 0
r = ~~1~~ ~~0~~ ~~1~~ 0
db = ~~1~~ 0

WWW.INDIAPUSTAK.com 9560127702

DINING PHILOSOPHERS

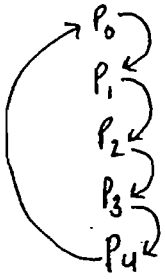
$\boxed{N = 5}$




```
void philosopher(int i)   → philosopher NO.
{
    while(true)
    {
        thinking();
        take-fork(i);                  // taking the left fork
        take-fork((i+1)% N);           // taking the right fork
        eat();
        put-fork(i);                   // put left fork back
        put-fork((i+1)% N);            // put right fork back.
    }
}
```

WWW.INDIANPUSTAK.com 9560127702

Note:
- if all the philosophers are hungry at the same time, then everybody will take their left fork first and when they try to attempt to take their right fork then nobody will get the right fork.
- All the philosophers will wait for each other and they will go into deadlock.



Correct Solution :

```
# define N 5
# define THINKING 0
# define HUNGRY 1
# define EATING 2
# define LEFT (i+N-1)%N
# define RIGHT (i+1)%N
```

→ left philosopher.

→ right philosopher.

```
Semaphore mutex = 1.
Semaphore s[N];
    //all s[i]'s are initialized to 0
int state [N];
    //an array to keep track of everyone state.

void philosopher(int i)
```

→ philosopher #

```
{ while (true)
    {  thinking();
       take_forks(i);
       eat ();
       put_forks(i);
    }
}
```

```
void take_forks (int i)
{ down (mutex);
  state[i] = HUNGRY ;
  test (i);
  up (mutex);
  down (s[i]);
}
```

```
void put_forks (int i)
{ down (mutex);
  state [i] = THINKING;
  test (LEFT);
  test (RIGHT);
  up (mutex);
}
```

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

WWW.INDIAPUSTAK.com 9560127702

- $\boxed{\text{mutex}}$ is a binary semaphore used by the philosopher in a mutual exclusive manner.
- $\boxed{S[N]}$ is an array of binary semaphore. Initially all are assigned to zero.
- $\boxed{\text{state}[N]}$ is an integer array used to keep track of every philosopher state. Initally all the philosopher's will be in the thinking state.

Analysis:

mutex = ~~1~~ 0 1

$P_0 = T$	$S[0] = 0$.
$P_1 = T$	$S[1] = 0$
$\Rightarrow P_2 = \cancel{T}\ \cancel{H}\ E$	$S[2] = \cancel{0}\ 1$
$P_3 = T$	$S[3] = 0$
$P_4 = T$	$S[4] = 0$

Ques. Assume that philosopher P_1 and P_3 are in the eating state, then philosopher P_2 is also trying to goto eating state. Then which stmt in the above code is controlling the philosopher to Not to go in the eating state

a) down(mutex)
b) if condition
c) test function
✓ d) down(s[i])

Analysis

mutex = ~~1~~ 0 1

$P_0 = T$	$S[0] = 0$
$P_1 = E$	$S[1] = 0$
$P_2 = \cancel{T}\ H$	$S[2] = 0$
$P_3 = E$	$S[3] = 0$
$P_4 = T$	$S[4] = 0$

WWW.INDIAPUTAK.com 9860127702

Ques. What happens if we interchange up(mutex), down(s[i]) in the take_forks function?

a). No prolem the sol^n still works correct.
b). ~~no~~ More than 2 philosophers can go into eating state at same time.
✓ c). It is possible for deadlock.
d). None of the above.

Analysis

mutex = 1

$P_0 = T$	$S[0] = \cancel{0}\ 1$
$P_1 = E$	$S[1] = \cancel{0}\ 1$
$P_2 = T$	$S[2] = \cancel{0}\ 1$
$P_3 = T$	$S[3] = \cancel{0}\ 1$
$P_4 = T$	$S[4] = \cancel{0}$

- Philosopher 2 wants to go in the eating state and preempt at down(s[i]);

Ques. Assume that philosopher P_1 is in the eating state. Then, philosopher P_2 is trying to go to eating state then it will be suspended. Then what is the procedure & how the philosopher P_2 will wake up & ~~going to eo~~ go into eating state.

Analysis

mutex = ~~X~~ 1

$P_0 = T$	$S[0] = 0$
$P_1 = E$	$S[1] = 0$
$P_2 = \cancel{T}\ H$	$S[2] = 0$
$P_3 = T$	$S[3] = 0$
$P_4 = T$	$S[4] = 0$

- Suspended because of down(S[i])
- when the P_1 stops Eating the P_2 will wakeup and eat his food.

24/11/2019

Ques. Let P_0 to P_4 be the processes and M[0] ..... M[4] be the binary semaphore mutexes all are initialized to '1'. Each process P_i executes the following code.

```
wait[M(i)];
wait(M[(i+1) % 4]);
[C.S]
signal(M[i]);
signal(M[(i+1)%4]);
```

Consider the below stmts.

I. ME is satisfied.

II. ME is not satisfied.

III. It is possible for deadlock.

Which of the above are true?

a). only I

b). only II.

c). only I, III

d). only II, III

Analysis

$P_0 = M_0, M_1$	$M_0 = \cancel{1}\ 0$
$P_1 = M_1, M_2$	$M_1 = \cancel{1}\ 0$
$P_2 = M_2, M_3$	$M_2 = \cancel{1}\ 0$
$P_3 = M_3, M_0$	$M_3 = \cancel{1}\ 0$
$P_4 = M_4, M_1$	$M_4 = 1\ 0$

[CS P_0 P_2]

WWW.INDIAPUSTAK.com 9560127702

Ques. Consider the following code used by the processes to access the common CS.

```
shared boolean lock = FALSE;
~~bot~~ boolean key;
do
  { key = TRUE;
    while (key == TRUE);
    swap (key, lock);
    [CS]
    lock = FALSE;
  } while (true);
```

Note: swap() function atomically swaps two values by using call by reference.

Consider the below stmts

I. ME is satisfied.

II. ME is not satisfied.

III. It is possible for deadlock.

Which of the above are TRUE?

a). only I

b). only II

c). only I, III

d). only II, III

WWW.INDIAPUSTAK.com 9560127702

Analysis

Lock = ~~FALSE~~ True

P_1	P_2	P_3
Key = ~~T~~ F	Key = T	Key = T

[C.S] P_1

IV. Programming Language Compiler Support Type

a) Monitors

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

Syntax:

```
keyword ← monitor example → Name of the monitor.
        {
          variables;
          condition variables;
          procedure p1
          [{ ==
            ==
            ==
          }
          procedure p2
          [{ ==
            ==
          }
        }
```

WWW.INDIAPUSTAK.com 9560127702

Condition Variables:

eg: $\boxed{\text{condition } x, y;}$

Two different operations performed on the condition variables of the monitor.

1. Wait();
2. Signal();

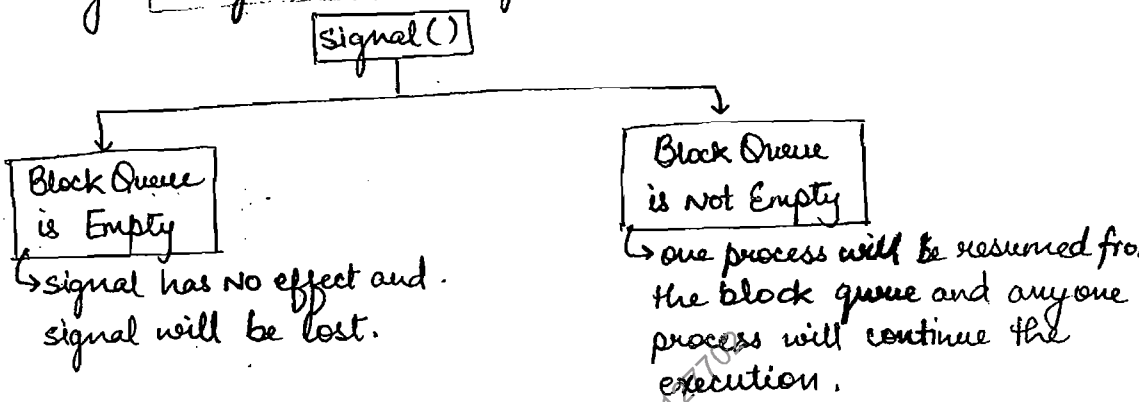
1). Wait()

eg : x.wait(); or wait(x);

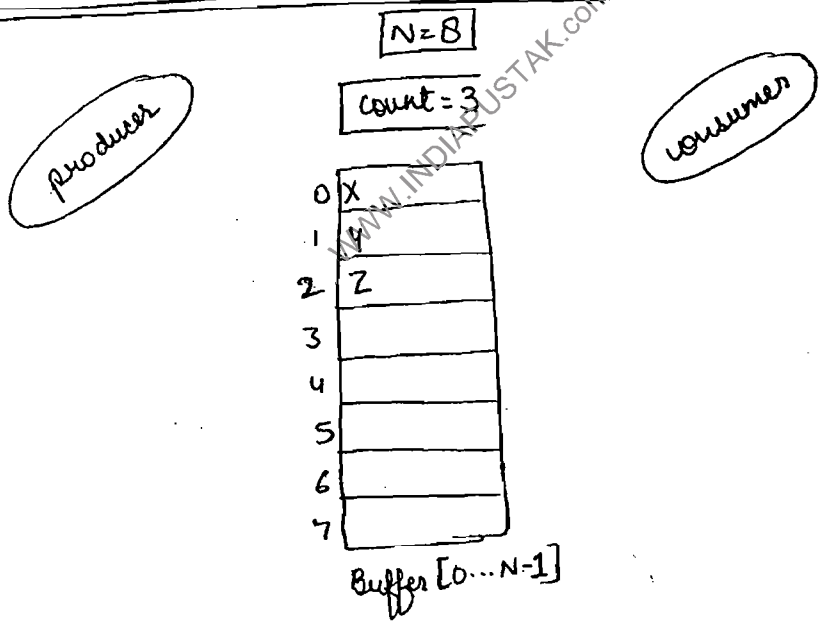
The process performing wait operation on any condition variable will be suspended and suspended process will be placed in the 'Block Queue' of respective condition variable.

2). Signal()

eg : x.signal(); or signal(x);



Producer-Consumer Problem with Monitors



WWW.INDIANOTESTAK.COM 9960127702

```
monitor producer-consumer
{
  int count = 0;
  condition Empty, Full;
  procedure Enter_item
  {
    if (count == N) wait (Full);
    enter (item);
    count = count + 1;
    if (count == 1) signal (Empty);
  }
  procedure Remove_item
  {
    if (count == 0) wait(Empty);
    remove (item);
    count = count - 1;
    if (count == N-1) signal(Full);
  }
}
```

```
procedure producer
{
  int itemp;
  while (true)
  {
    produce_i
    producer_consumer.Enter_items;
  }
}
```

```
procedure consumer
{
  int itemc;
  while (true)
  {
    produce_consumer.Remove_item;
    process_item (itemc);
  }
}
```

WWW.INDIAPUSTAK.com 9560127702

Analysis:

Buffer is Full.

Block Queue of Full

producer

Buffer is Empty.

Block Queue of Empty

consumer

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

5 CONCURRENT PROGRAMMING

S_1 : a = b + c ;

S_2 : d = e * f ;

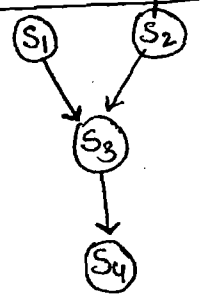
S_3 : g = a / d ;

S_4 : h = g * i ;

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

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

· precedance Graph



Note: 1. Any two statements 'S_i' and 'S_j' can be executed concurrently and parallely if they follow below conditions.

a). $R(S_i) \cap W(S_j) = \phi$

b). $W(S_i) \cap R(S_j) = \phi$

c). $W(S_i) \cap W(S_j) = \phi$

2. The real concurrent programming is possible only on the multiprocessor system

3. The concurrent has different meanings

a). They don't have any dependency.

b). They can execute concurrently or parallely.

c). Anyone can start first.

4. The concurrent programming will be written by using.

parallel ← par begin – par end.

OR

concurrent ← co begin – co end.

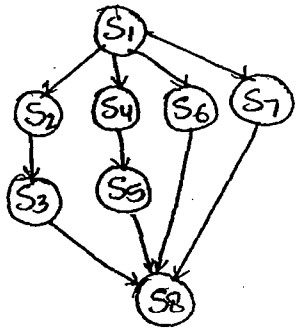
```
begin
S1 ;
S2;
S3 ;
end
```

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

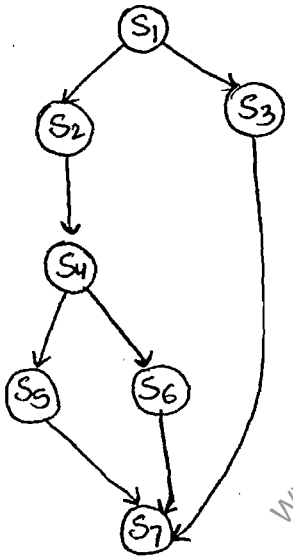
WWW.INDIAPUSTAK.com 9560127702

Exp:

```
S1;
par begin
  begin
    S2;
    S3;
  end.
  begin
    S4;
    S5;
  end
  S6;
  S7;
par end.
  S8;
```



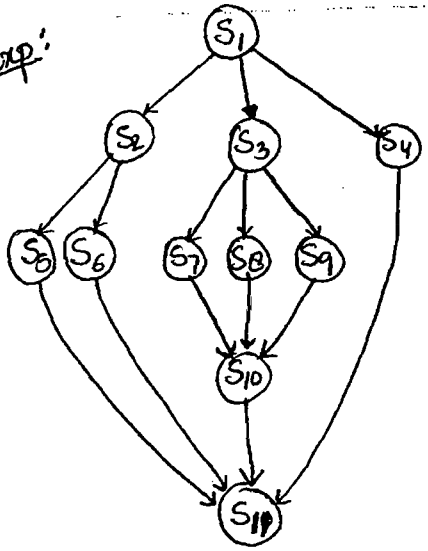
Exp:




```
S1;
par begin
  begin
    S2;
    S4;
    par begin
      S5;
      S6;
    par end;
  end
  S3;
par end.
S7;
```

WWW.INDIAPUSTAK.com 9560127700

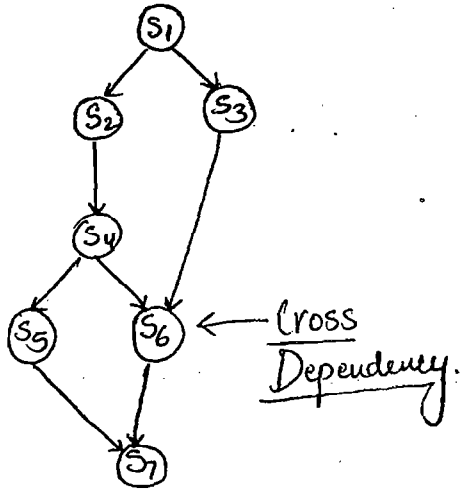
Exp:




```
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;
```

Note: It is not possible to write concurrent program for all the precedence graphs by using par begin & par-end but it is very much possible with the help of semaphore operations.

Exp:



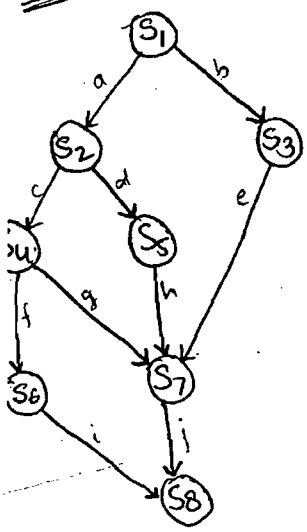

```
S1
par begin
  begin
    S2;
    S4;
    par begin
      S5;
      S6 X
```

All binary semaphores variables a, b, c, d, e, f, g, h are initialised '0'.

```
par begin
  begin S1, par begin v(a), v(b), par end, end;
  begin p(a), S2, v(c), end;
  begin p(b), S3, v(d), end;
  begin p(c), S4, par begin, v(e), v(f), par end, end;
  begin p(e), S5, v(g), end;
  begin p(d), p(f), S6, v(h), end;
  begin p(g), p(h), S7, end;
par end.
```

WWW.INDIABUSTINK.com 9560127702

Exm:




```
par begin
  begin S1, par begin v(a), v(b), par end, end;
  begin p(a), S2, par begin v(c), v(d), par end, end;
  being p(b), S3 v(e) end;
  begin p(c), S4 par begin v(f), v(g), par end, end;
  begin p(d), S5, v(h), end.
  begin p(f), S6, v(i), end.
  begin p(e), p(g), p(h) S7, ~~end~~ v(j) end.
  begin p(i), p(j) S8, end;
par end.
```

Ques.

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

What can be the value of x and y after completion of the above concurrent program?

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

Which of the above claims are possible?

a). Only I, II
b) only II, III
c) only I, III
d). ~~and~~ All I, II, III

Q.25 (82).

W, X	Y, Z
P(s)	P(s)
I. Load Ri, M[X]	I. Load Ri, M[X]
II. INCR Ri	II. DECR Ri
III. Store M[X], Ri	III Store M[X], Ri
V(s);	V(s);

W → Y → Z → X

W	X	Y	Z
p(s);	p(s);	p(s);	p(s);
I. Load Rw, M[X]	I Load Rx, M[X]	I. load Ry, M[X]	I. load Ry, M[X]
II. Incr Rw	II. Incr Rx	II. Decr Ry	II. Decr Rz
III Store M[X], Rw	III Store M[X], Rx	III. Store M[X], Ry	III Store M[X], Rz
V(s);	V(s);	V(s);	V(s);
+1	+2	-2	-4

Ans. +2.

Analysis

x = ∅ - 2 - 4 × 2

W → I
W → II

Y → I II III
Z → I II III

W → III
X → I II III

WWW.INDIAPUSTAK.com 9560127702

Ques. Consider the following concurrent program.

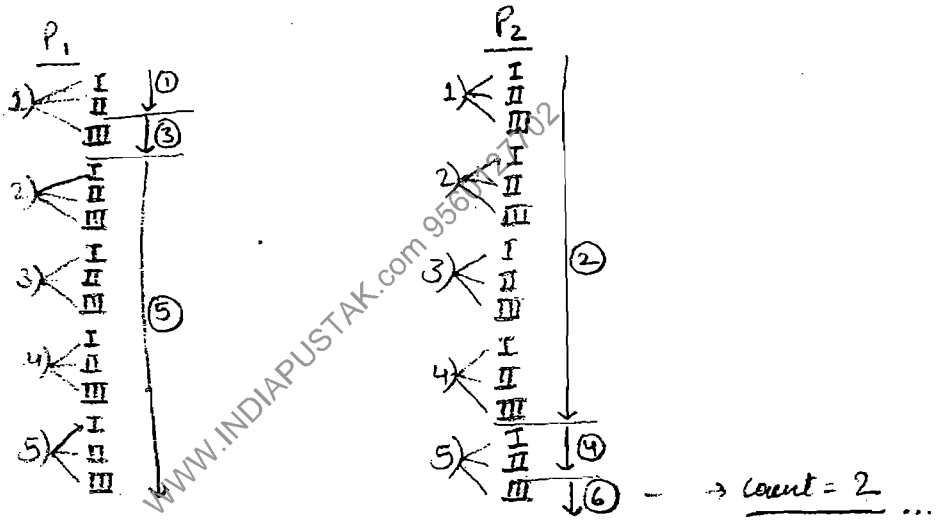
```
int count = 0;
void tally()
{
  int i;
  for (i=1; i<5; i++)
  count = count + 1;
}
main()
{
  parbegin
      tally(); — P1
      tally(); — P2
  parend.
}
```

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

Note: count = count + 1, will execute 3 instructions like load, increment and then store.

I. Load Ri, M[count]
II. Incr Ri
III. Store M[count], Ri

a). 1 ✓b). 2 c). 3 d). 4 e). 5



WWW.INDIAPUSTAK.com 9561172702

Analysis.

count = ~~0~~ ~~1~~ ~~1~~ ~~0~~ 2

P_1 : 1st iteration → I R_1 [~~0~~ 1]
P_1 : 1st iteration → II

P_2 : 1, 2, 3, 4 iterations.

P_1 : 1st iteration → III

P_2 : 5th iteration – I R_2 [~~1~~ 2]
P_2 : 5th iteration – II.

P_1 : 2, 3, 4, 5 iterations

P_2 : 5th iterations – III.

CPU Scheduling [W.B.]

Q.(11). TQ = Q
C.S = T
I/O = R

$$\text{CPU Efficiency} = \frac{\text{Useful Time spent by CPU}}{\text{Total Time spent by CPU}} = \boxed{\frac{Q}{Q+T}} = \frac{Q}{Q+T+R}$$

I/O →
1). I/O may occur alone
2). I/O may occur along with CPU.
3). Multiple I/Os may occur simultaneously.

for Q.32 and Q.33

In the shortest Job first scheduling algorithm, the burst time of the processes are expected by using the following formula.

$$\boxed{\tau_{n+1} = \alpha \cdot T_n + (1-\alpha)\tau_n}$$

where τ_{n+1} = Next expected Burst Time.
τ_n = previous expected burst time.
T_n = previous actual burst time.
α = a parameter which controls relative weight of recent & past history.

$$\boxed{0 \le \alpha \le 1}$$

WWW.INDIAPUSTAK.COM 9560127702

Q(32).

$\tau_1 = 10$
$\alpha = 0.5$
$T = 5$

P	BT
P_1	9
P_2	8
P_3	3
P_4	5

$\tau_2 = 0.5 * 5 + 0.5 * 10 = 7.5$

$\tau_3 = 0.5 * 8 + 0.5 * 7.5 = 7.75$

$\tau_4 = 0.5 * 3 + 0.5 * 7.75 = 5.375$

$\tau_5 = 0.5 * 5 + 0.5 * 5.375 = \underline{5.1875}$ Ans (a).

Q.(33).

$\tau_5 = \alpha \cdot T_4 + (1-\alpha)\tau_4$

$= 1 * 5 + \underline{0}$

$= \underline{5}$ Ans (c).

25/11/2019

DEADLOCKS

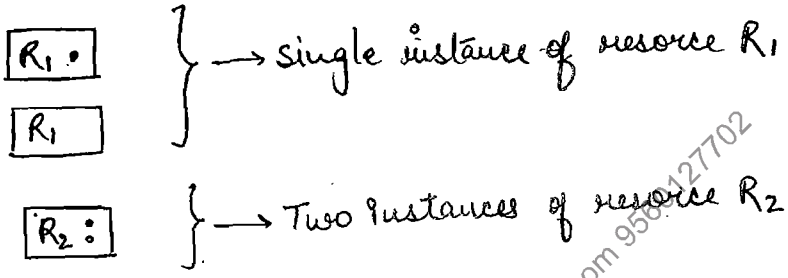
Defination: The two or more processes are waiting on some event to happen, which never happens, then those processes are said to be involved in the deadlock.

- **Basics of Deadlocks:**

processes: P_1 P_2

Resources: R_1 R_2

Resources with Instances:

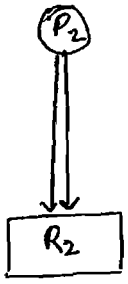


R_1, R_1 → single instance of resource R_1

R_2 → Two instances of resource R_2

Requesting Edge

process 'P_1' is requesting for '1' instance of Resource 'R_1'



process 'P_2' is requesting for '2' instances of Resource R_2

Allocation Edge

'1' instance of Resource 'R_1' is allocated to process 'P_1'

'2' instances of Resource R_2 is allocated to process 'P_2'

Note: • The resource request & resource allocation will be represented by using resource allocation graph.

WWW.INDIAPUSTAK.com 9580127702

- Resource Allocation Graph contains vertices & Edges.

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

V → processes, Resources.

E → Requesting, Allocation

The Resource Request / Release Life Cycle:

1). The process will make the request for the resources.

2). The operating system clearly validates the request of the process.

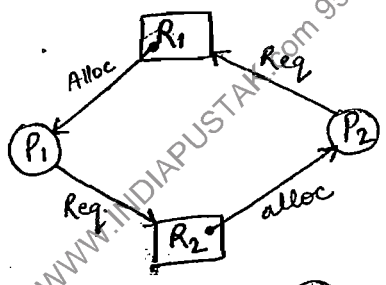
3). If the request made by the process is valid, then operating system will check for the availability of the resource.

4). If the resource is freely available then it will be allocated to the process, otherwise process has to wait.

5). If all the resources required for the start of execution are allocated, then the process will go into execution.

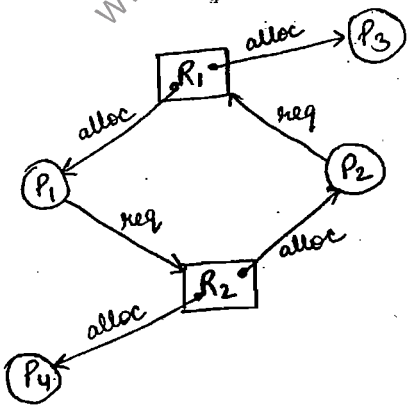
⊛6). Once the execution of the process is completed, then the process will release all the resources.

Eg:



Deadlock exists in the R.A.G.

Eg:



No Deadlock exists in the R.A.G.

Ques. Consider a system which has 'n' ~~CPU~~ processes and 6 Tape Drives. Each process requires 2 tape drives to complete their execution then what is the maximum value of N. which ensures deadlock free execution.

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

P_1	P_2	P_3	P_4	P_5
1	1	1	1	1
1				
1				

← worst case

www.INDIAPUSTAK.com 9560127702

Ques. Consider a system which has 3 processes and each process requires 2 resources to complete their execution. Then what is the minimum no. of resources required to ensure deadlock free execution.

a). 2
b) 3
c) 4 ✓
d) 5

P_1	P_2	P_3
1	1	1
1		

Ques. Consider a system which has 'n' processes and 6 resources. Each process requires 3 resources to complete their execution. Then what is the max. value of N which ensures deadlock free execution?

a) 2 ✓
b) 3
c) 4
d) 5

P_1	P_2
1	1
1	1
1	1

Ques. Consider a system which has three processes P_1 P_2 and P_3 The peak demand of each process is 5, 9, 13 respectively. for the Resource R. Then what is minimum No. of resources required to ensure deadlock free execution. 25 ?

$P_1 \rightarrow 5 \rightarrow 4$
$P_2 \rightarrow 9 \rightarrow 8$
$P_3 \rightarrow 13 \rightarrow 12$

$24 + 1 \Rightarrow 25$

WWW.INDIAPUSTAK.com 9560127772

* In the Deadlock we study 2 things:

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

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

→ The least upper bound, which satisfies the condition is called as Sufficient condition.

Q 13 / 97

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

$P_1 = 9$
$P_2 = 9$
⋮
$P_{100} = 9$

$m = 10$
$9 < 10$

(b) $\forall i, S_i < n$

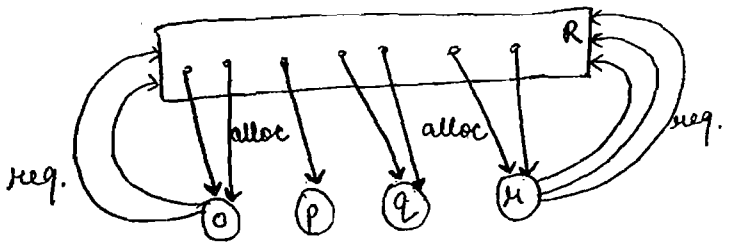
(c) $\sum_{i=1}^{n} S_i - n + 1 = m$

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

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

Q.10
(96)

$n = 4(0, p, q, r)$.



$x_0 = 2$ $\quad$ $y_0 = 2$

$x_p = 1$ $\quad$ $y_p = 0$ $\quad$ $\boxed{y_p = y_q = 0}$

$x_q = 2$ $\quad$ $y_q = 0$

$x_r = 2$ $\quad$ $y_r = 3$

$\boxed{x_p + x_q \geq \text{Max } y_k}$ $\quad k \neq p, q$.

available resources.

Concepts of Deadlocks :-

1. Deadlock Characteristics
2. Deadlock prevention
3. Deadlock avoidance
4. Deadlock detection.
5. Deadlock recovery.

WWW.INDIAPUSTAK.com 9560127702

(1). Deadlock Characteristics :

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

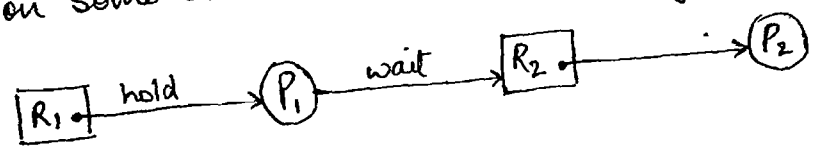
• There should be a one to one relationship b/w a ~~resource~~ resource and a process.

2). No Preemption : • The resource has to be 'volantarily' released by the process after completion of the execution.

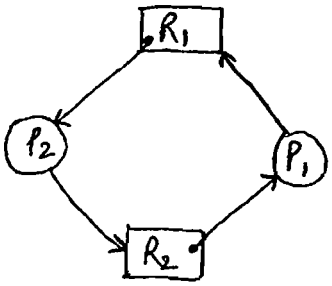
• It is not allowed to forcefully preempt the resource from the process.

3). Hold and Wait : • The process is holding the resource and waiting on some other resource simultaneously.

•



4). **Circular Wait** : • The processes are circularly waiting on each other for the resources.



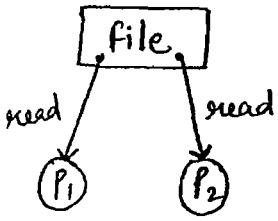
$\rightarrow R_1 \rightarrow P_1 \rightarrow R_2 \rightarrow P_2 \rightarrow$

Note: If all the above 4 conditions are existing simultaneously in the system, then definately there exist a deadlock.

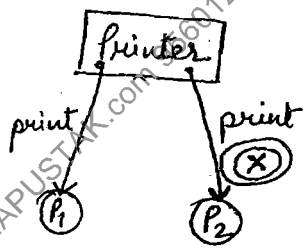
(2.) Deadlock Prevention:

we try to dissatisfy the above (4) conditions

1). **Mutual Exclusion**: It is not possible to dissatisfy the mutual exclusion always because of sharable & Non sharable resources.



Sharable Resources



Non Sharable Resources

WWW.INDIAPUSTAK.COM 9560127702

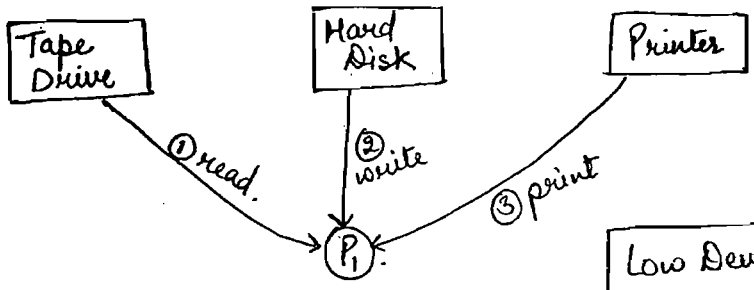
2). **No preemption**:

The process 'P_1' is requesting for resource 'R_1'

- If the resource 'R_1' is free, then it will be allocated to process P_1.
- The resource 'R_1' is not free and it is allocated to some other process 'P_2'.
  - If the process 'P_2' is in the execution, then process 'P_1' has to wait
  - The process 'P_2' is not in the execution & it is waiting on some other resource 'R_2'
    - ↳ preempt resource 'R_1' from process P_2 and allocate to the process P_1.

3). Hold and Wait :

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



Low Device Utilization
(OR)
Less Resorce Utilization

(ii) The process should release, all the existing resorces before making the New request.

1). $P_1 \xrightarrow{\text{Req. 100}} \boxed{R}$, $\boxed{R} \xrightarrow{\text{alloc 75}} P_1$ release 75

2). $P_1 \xrightarrow{\text{Req. 100}} \boxed{R}$, $\boxed{R} \xrightarrow{\text{alloc 65}} P_1$ release 65

⋮

Starvation

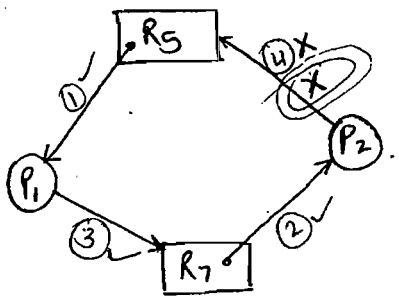
WWW.INDIAPUSTAK.COM 9560127702

4). Circular Wait :

- Resorces will be assigned with the unique Numerical Numbers.
- The process can request for the resorce only in the increasing (or decreasing) order of Enumeration (numbering).

Eg: $P_1 \longrightarrow R_5$ ✓ | $P_1 \longrightarrow R_3$ ✗ | $P_1 \longrightarrow R_7$ ✓

Eg:



Note: By dissatisfying any of the above ④ conditions, we can prevent the deadlock.

(3). Deadlock Avoidance :

Deadlock avoidance is implemented by using **Banker's algorithm.**

BANKER's Algorithm

Exp:

	Max Need			Current allocation			Current available			Remaining Need		
	A	B	C	A	B	C	A	B	C	A	B	C
$P_0 \rightarrow$	7	5	3	0	1	0	3	3	2	7	4	3
$P_1 \rightarrow$	3	2	2	2	0	0	5	3	2	1	2	2
$P_2 \rightarrow$	9	0	2	3	0	2	7	4	3	6	0	0
$P_3 \rightarrow$	2	2	2	2	1	1	7	5	3	0	1	1
$P_4 \rightarrow$	4	3	3	0	0	2	10	5	5	4	3	1
				7	2	5	10	5	7			

Safe Sequence ⇒ $P_1 P_3 P_0 P_2 P_4$

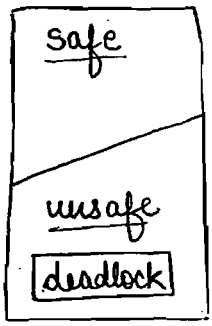
Total Available		
A	B	C
↓	↓	↓
10	5	7

processes ⇒ P_0 P_1 P_2 P_3 P_4
Resources ⇒ A, B, C.

Remaining Need = Max Need − Current Allocation

Current Available = Total available − Total Current allocation

WWW.INDIAPUSTAK.COM 9560127702



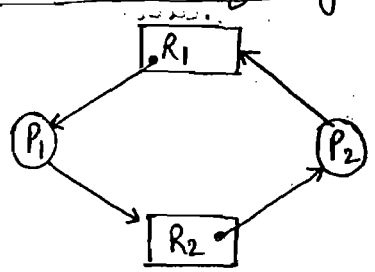
- if we can satisfy remaining need of all the processes with the current available resources, then the system is said to be in the safe state. Otherwise system is said to be in the unsafe state.
- If the system is in the unsafe state, then it is possible for the deadlock.
- The order in which we satisfy the remaining need of all the processes is called as "Safe Sequence."
- Safe Sequence may not be unique. There can be multiple safe sequences.
- The unsafe state purely depends on behaviour of the processes.
- The deadlock Avoidance is less restrictive than the deadlock prevention.

Note: Each and every time when the process is requesting for any resource, the banker's algorithm will be implemented to identify whether the system is in the safe state or in the unsafe state.

- If the system is in the safe state then the request of the process will be granted and if the system is in the unsafe state then the request of the process will be denied and deadlock will be avoided.

(4) Deadlock Detection:

(i) The resources are of single instance type.



Run: Cycle Detection Algorithm.

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

cycle.

- If all the resources are of single instance type, then cycle in the resource allocation graph is necessary & sufficient condition for occuring of deadlock.
- If all the resources are of Not single instance type, then cycle in the resource allocation graph is just a Necessary condition but Not sufficient condition for occuring of the deadlock.

(ii) The Resources are of multiple instance type.

	current allocation			current available			Remaining Need		
	A	B	C	A	B	C	A	B	C
$P_0 \rightarrow$	0	1	0	0	0	0	0	0	0
$P_1 \rightarrow$	2	0	0	0	1	0	2	0	2
$P_2 \rightarrow$	3	0	3	3	1	3	0	0	0
$P_3 \rightarrow$	2	1	1	5	2	4	1	0	0
$P_4 \rightarrow$	0	0	2	5	2	6	0	0	2
				7	2	6			

Safe State

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

WWW.INDIAMUSTAK.com 9560127702

Q. What happens if the process P_2 is requesting for additional resources of (0,0,1).

```
  000 (old)
+ 001 (new)
  001 (latest)
```

	current allocation				current available				Remaining Need		
	A	B	C		A	B	C		A	B	C
$P_0 \rightarrow$	0	1	0	$\xrightarrow{+}$	0	0	0		0	0	0
$P_1 \rightarrow$	2	0	0		0	1	0		2	0	2
$P_2 \rightarrow$	3	0	3						0	0	1
$P_3 \rightarrow$	2	1	1						1	0	0
$P_4 \rightarrow$	0	0	2						0	0	2

$P_0 \Rightarrow$ Unsafe $\Rightarrow$ Deadlock P_1, P_2, P_3, P_4

5. Deadlock Recovery

1). Killing the process :

(i). Kill all the processes which are involved in the deadlock.

(ii) Kill one after the other.

→ Low priority process

→ % process completion.

$P_1 \rightarrow 90\%$ $\quad$ $P_2 \rightarrow 10\%$ ✓

→ # of resources the process is Holding.

P_1 ✓ ← Hold ← 20R $\quad$ P_2 ← Hold ← 1R

→ # of resources the process is requesting.

$P_1 \xrightarrow{\text{req. } 20} R$ $\quad$ $P_2 \xrightarrow{\text{req } 1.} R$

WWW.INDIABUSTAK.com 9560127702

2). Resource Preemption :

- Resources will be preempted from the processes which are involved in the deadlock & preempted resources will be allocated to some other processes, so that their may be a possibility to recover the system from deadlock.
- If the above condition is followed then there may be a possibility for the process to go into 'Starvation'.

3). Ostrich Algorithm :

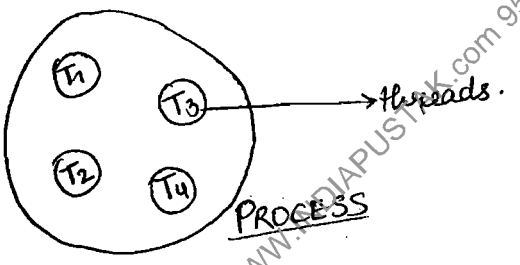
→ ignore the deadlocks.

THREADS

Defination: The light weight process.

→ 1). No. of Instruction will be less.

→ 2). Context will be Less.

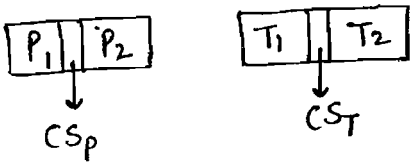


WWW.INDIAPUSTAK.com 9560127702

Advantages of Threads:

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

2). Faster Context Switches :



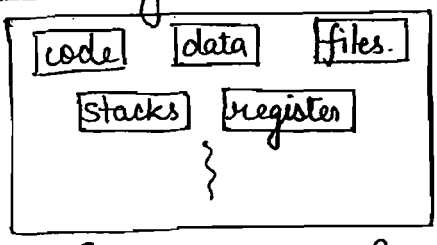
$CS_T << CS_P$

The context switching time between the threads will be very less compared to the context switching time between the processes because Threads will have less context compared to the Process.

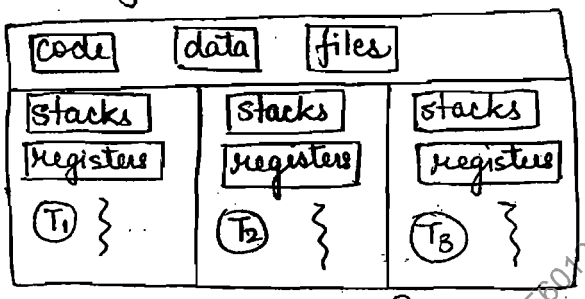
3). Effective Utilization of Multi Processor Systems :

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

4). Resource Sharing :



Single Threaded Process.



Multi Threaded Process.

WWW.INDIANUSTA.com 9560127702

The resorces like code, data (global variables), files, heap and memory will be shared among all the threads within the process but Stacks & Registers cannot be shared.

5). Enhanced Throughput of the Systems :

If the process is divided into multiple threads & if we consider one thread as one Job then no. of Jobs completed per unit time will increase & throughput of the system will be enhanced.

6). Economical :

Implementation of the threads does not required any cost. and there are various programming language. API's which supports implementation of the threads.

eg: JAVA API.

Threads are further categorized into two types :

1. User Level Thread
2. Kernel Level Thread.

26/11/2019

User Level Threads	Kernel Level Threads.
1. User level threads are implemented by user & programmer.	1. Kernel Level Threads are implemented by operating system.
2. Operating System cannot recognise user level thread and it use user level thread as process only.	2. Kernel Level Threads are recognised by operating system.
3. If one user level thread is performing blocking system call then entire process will go into block state.	3. If one kernel level thread is performing I/O operation then another thread will continue the execution.
4. User level threads are designed as dependent Threads.	4. Kernel Level Threads are designed as independent Threads.
5. Implementation of User Level Threads is easy or simple.	5. Implementation of Kernel Level threads is complicated or difficult.
6. User Level Threads will have less context.	6. Kernel Level Threads will have more context.
7. No hardware support require for the user level threads.	7. Scheduling of Kernel Level Threads requires hardware support.

Note: "GANG Scheduling" is used in the Threads.

~~I/O of the process~~

I/O Operations:

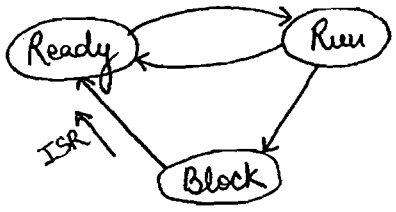
- I/O of the process is categorized into two types

1. Synchronous I/O
2. Asynchronous I/O

WWW.INDIANPUSTAK.com 9560127202

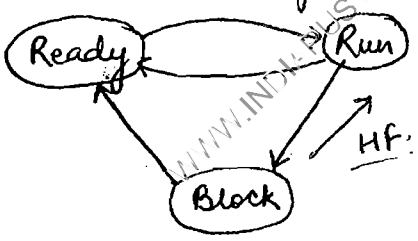
Synchronous I/O :

- In Synchronous I/O process performing IO operation will be placed in the block state till the IO operation is completed.
- Once IO operation is completed, an ISR will be initiated which brings the process from block state to ready state.



Asynchronous I/O :

- In asynchronous I/O, while initiating the I/O request the ~~handler~~ "~~Hod~~ "Handler Function" will be registered.
- The process is Not placed in the block state and it continues to execute remaining code after initiating the I/O request
- Once I/O operation is completed, the signal mechanism is used to notify the process that the data is available. and registered Handler function asynchronously invoked



WWW.INDIAPLUS.IN.com 9568327712

Chapter-3 — William Stallings

Memory Management

Functionality: Allocating and deallocating the memory to the processes.

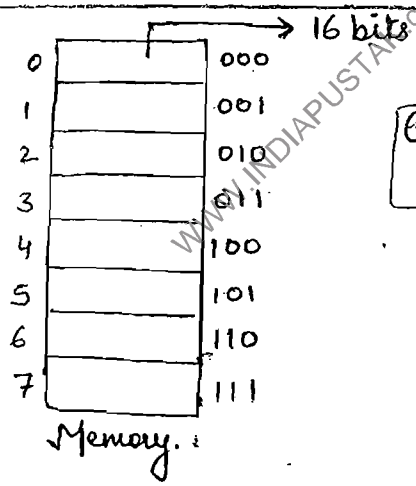
Goal: The efficient utilization of memory by minimizing the internal & external fragmentation.

Main Memory
Primary Memory
Physical Memory
RAM

Memory will be represented in 2 different ways:

Memory → Byte Addressable eg. 512KB
Memory → Word Addressable. eg. 512KW.

$2^2 = 4$	
$2^3 = 8$	$2^{10} = 1024 \approx 10^3 = 1K$
$2^4 = 16$	$2^{20} = 1M$
$2^6 = 64$	$2^{30} = 1G$
$2^8 = 256$	$2^{40} = 1T$
$2^9 = 512$	



Capacity of the Memory = # words in memory * word size
= 8 * 16 bits
= 8 * 2B
= 16 B.

Ques. Consider a system which has 512G words and each word is having size of 128 bits. What is the capacity of memory in Bytes.

Capacity of the memory. = 512G * 128 bits
= 512G * 16 Bytes.
= 8TB

$2^9 * 2^{30} * 2^4$ B

Ques. Consider a system which has 64M words and each word is having size of 48 bits then what is the capacity of memory in Bytes?

Capacity of the memory = 64M * 48 bits
= 64M * 6B.
= 384MB

WWW.INDIAPUSTAK.com 9568127702

Ques. Consider a system where 32 binary bits are used to represent all the words of the memory. and each word is having size of 32 bits then what is the capacity of the memory in Bytes?

$$\text{Capacity of the memory} = 2^{32} * 32 \text{ bits}$$
$$= 2^{32} * 4B$$
$$= \underline{16 GB}.$$

RAM CHIP IMPLEMENTATION:

Ques. Ram chip size = 128 B

Organise the MM capacity of 16 KB.

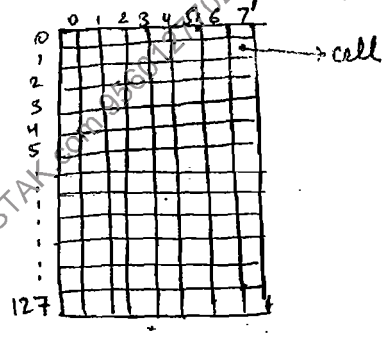
Q.1. # of RAM chip required.

Q.2. Draw the memory organisation map?

Q.3. What is the size of decoder required?

Q.4. Draw the memory organisation ~~map~~ map with the word size of 16 bits?

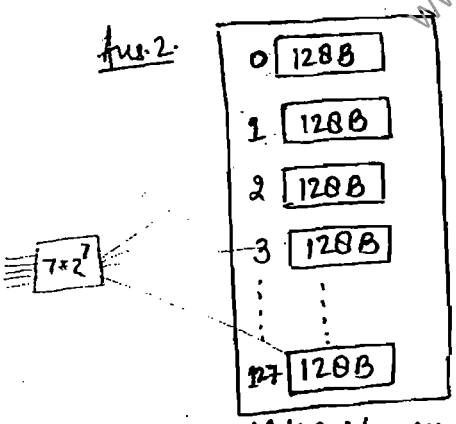
Ram chip size = 128 B.



Ans.1. # RAMchip required.

$$= \frac{16KB}{128B} = 2^7$$
$$= \underline{128}.$$

Ans.2.

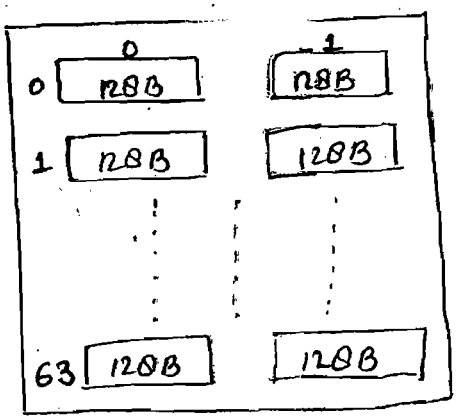


16 KB Memory, with word size as 8 bits.

Ans.3. Size of Decoder

$$\Rightarrow \boxed{7 \times 2^7}$$

Ans.4.



16 KB Memory, with word size as 16 bits.

WWW.INDIAPUSTAK.COM 9560027202

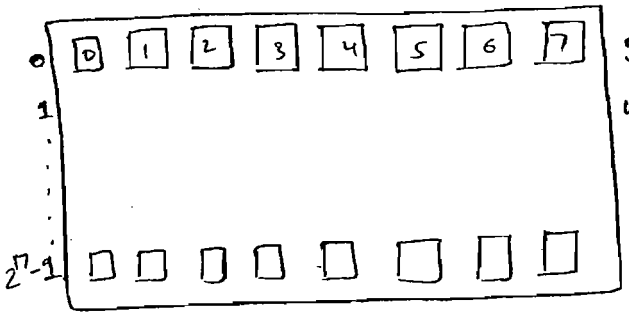
Ques. Ram chip size = 256 * 1 - Bit
Organise MM capacity of 32MB.

Ans.1. # RAM chip required

$$= \frac{32M * 8 \text{ bits}}{256 * 1 \text{ bit}}$$

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

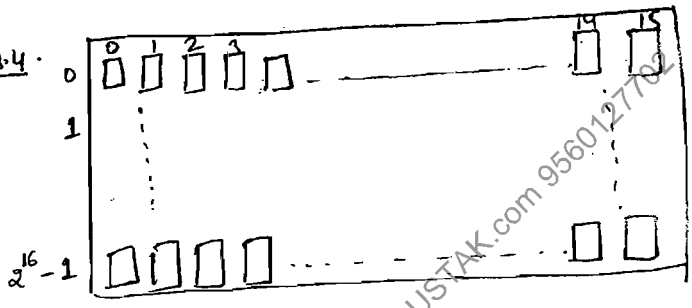
Ans.2.



32MB of memory with word size 8 bits.

Ans.3. $\boxed{17 * 2^{17} \text{ Decoder}}$

Ans.4.



32MB of memory with word size 16 bits.

WWW.INDIAPUSTAK.com 9560127702

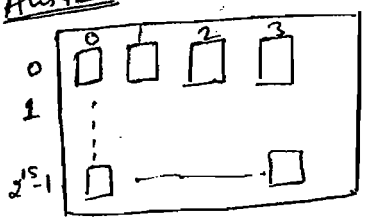
Ques. RAM chip size = 512 * 2 bits.
Organise the MM capacity of 16MB.

Ans.1. # RAM chip required.

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

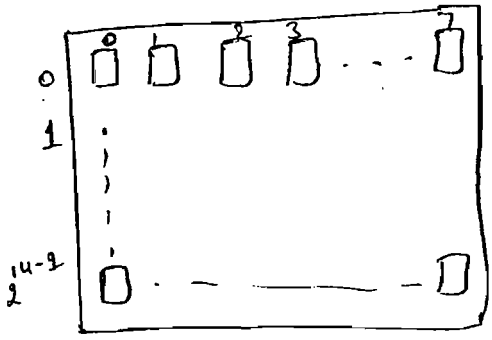
$$= 2^{17} = 128 K$$

Ans.2



Ans.3. $\boxed{15 \times 2^{15} \text{ Decoder}}$

Ans.4.



LOADING, LINKING & ADDRESS BINDING

Loading: Bring the program from secondary memory to MM is called as 'Loading'

Linking: Establishing the linking b/w all the modules or all the functions of the program in order to continue the program execution is called as Linking.

```
main()
{
  int x;
  x = add();
  ...
}

int add()
{
  ...
  return total;
}
```

Linking

Loading & Linking is categorised into 2 types:

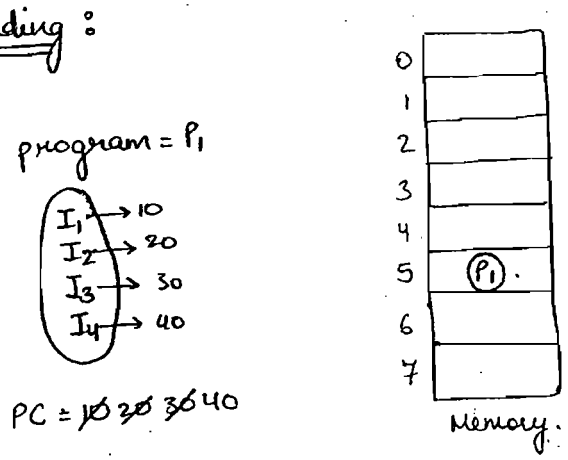
- Static
- Dynamic

WWW.INDIAPOSTAK.COM 9560122702

Static:
- Loading the entire program into memory before start of program execution is called as Static Loading.
- Inefficient utilization of the memory.
- The program execution will be faster.
- If the static loading is used accordingly, the static linking will be applied.

Dynamic:
- Loading the program into memory on demand is called as dynamic loading.
- Efficient utilization of memory.
- The program execution will be slower.
- If the dynamic loading is used, then accordingly dynamic linking will be applied.

Address Binding :



- Association of program instructions & data to the actual physical memory location is called as address binding.

Address Binding is categorized into 3 types :

1. Compile Time A.B
2. Load Time A.B
3. Execution Time A.B or Dynamic A.B

WWW.INDIAPUSTAK.COM 9960127702

(1) Compile Time Address Binding :

- If the compiler is responsible of performing address binding then, it is called as compile time A.B.
- This type of A.B will be done before loading the program into memory.
- The compiler requires to interact with the operating System memory manager to perform compile time A.B.

(2) Load Time Address Binding :

- This type of A.B will be done by operating system memory manager (Loader)
- This type of A.B will be done after loading the program into memory.

(3) Execution Time A.B or Dynamic A.B :

~~This typ~~

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

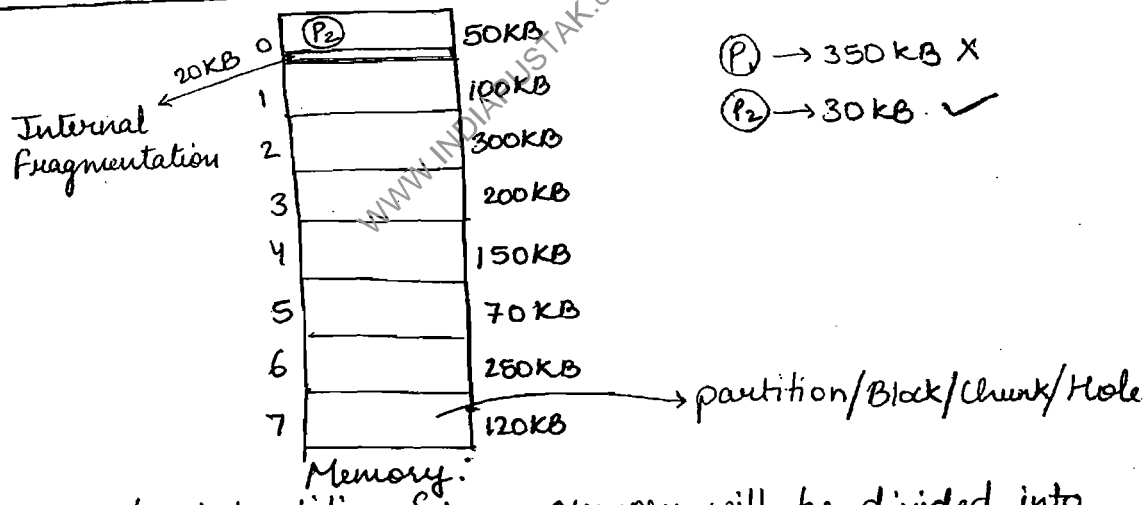
Note: Majority of the operating system practically implement dynamic loading, dynamic linking and dynamic address binding.
Eg: Windows, Unix, Linux etc..

MEMORY MANAGEMENT TECHNIQUES

Contiguous
- 1. Fixed Partition Scheme
- 2. Variable Partition Scheme.

**** Non-Contiguous**
- 1. Paging
- 2. Multi level Paging
- 3. Inverted Paging.
- 4. Segmentation
- 5. Segmented Paging

FIXED PARTITION SCHEME

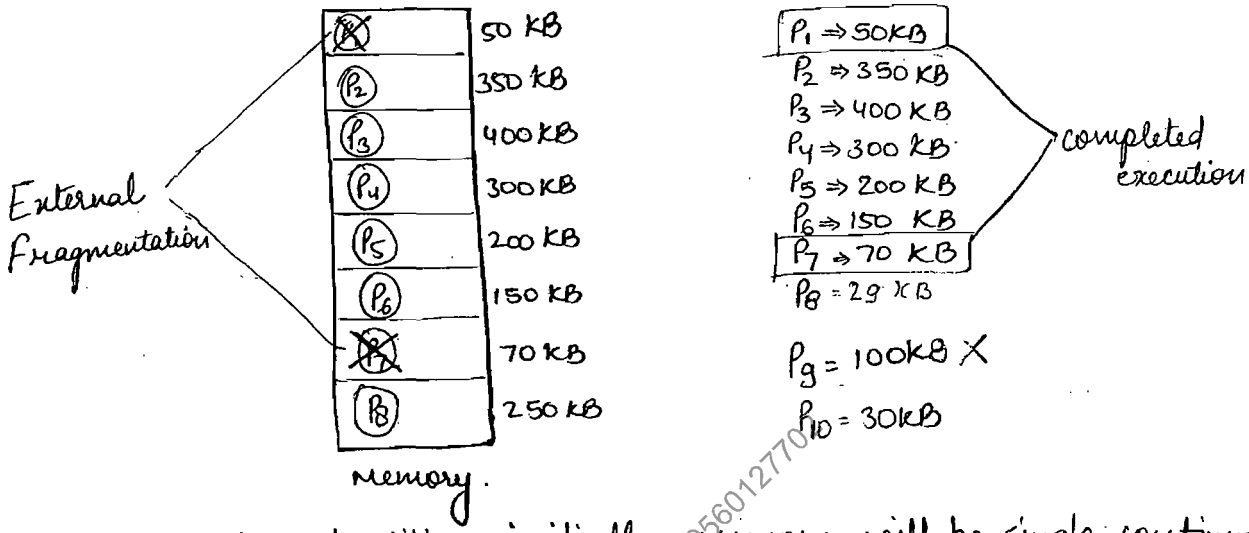


- In the fixed partition Scheme, Memory will be divided into fixed Number of partitions.
- Fixed means Number of partitions are fixed in the memory.
- In the fixed partition scheme, in every partition, only one process will be accomodated.
- Degree of Multiprogramming is restricted by No. of partitions in the memory.
- Maximum size of the process is restricted by maximum size of the partition.

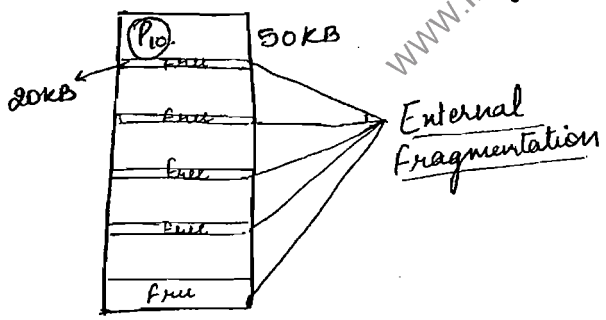
WWW.INDIANHUSTAK.com 9560127702

- Every partition is associated with the <u>limit registers</u>.
- Limit registers are categorised into 2 types.
  - → 1. <u>Lower Limit</u> : starting address of the partition.
  - → 2. <u>Upper Limit</u> : Ending address of the partition.

VARIABLE PARTITION SCHEME



- In the variable partition, initially memory will be single continuous free block.
- Whenever the request by the process arrise, accordingly partition will be made in the memory.



- If the smaller processes keep on coming, then the larger partition will be made into smaller partitions.
- To avoid the problem of External Fragmentation, following Techniques are used.
  1. Compaction.
  2. Implementing Non-contiguous M.M.T.
     - → paging.
     - → Segmentation.
     - etc..

WWW.INDIAPLUSTAK.com 9560127700

1). Compaction : Moving all the process towards the top or towards the bottom to make the free available memory in a single continuous place is called as Compaction.

Compaction is undesirable to implement because it interrupts all the running processes in the memory.

2). Implementing Non-contiguous Memory Mgmt Techniques :

Partition Allocation Methods

1). FIRST FIT : Allocate the process in a partition which is a first sufficient partition from Top of the memory.

2). BEST FIT : Allocate the process in a partition which is smallest sufficient among the free available partition.

To find out the smallest sufficient, it requires to search all the free partitions in the memory.

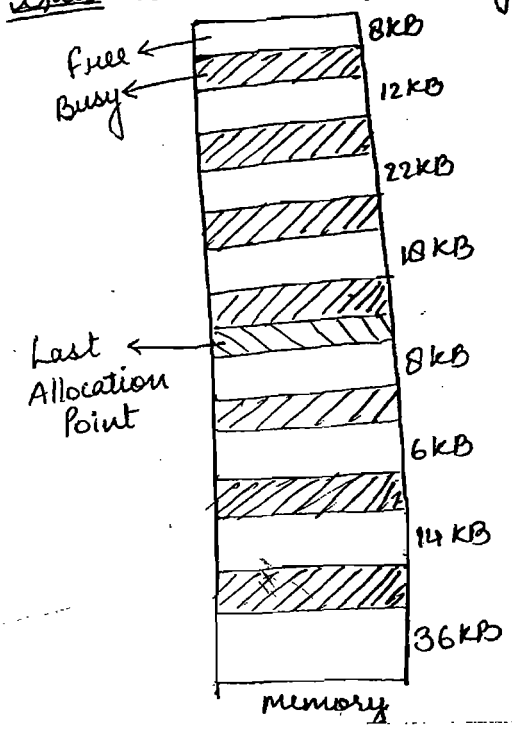
3). WORST FIT : Allocate the process in a partition which is largest sufficient among the free available partition.

To find out the largest sufficient, it requires to search all the free partitions in the memory.

4). NEXT FIT : Next fit also works like first fit. But it will search for the first sufficient partition from "LAST allocation Point".

WWW.INDIAPRESTIGE.com 9566127702

Ques. Consider the following Memory map

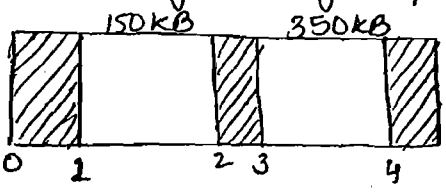


Q.1. for P_1 = 16KB partition alloc.

(i) First fit : 22KB
(ii) Best fit : 18KB
(iii) Worst fit : 35KB
(iv) Next fit : 36KB

Q.2. How many successive requests of 6KB will be satisfied, by using first fit assuming variable partition Scheme. 19. ?

Ques. Consider the following Memory Map.



The requests from the processes are 300 KB, 25 KB, 125 KB, 50 KB respectively. The above requests will be satisfied with? (assume variable partition scheme)

a). Best fit but Not First fit
b) First fit but Not Best fit ✓
c) Both first fit & best fit
d). Neither first fit NOR Best Fit.

27/11/2019

PAGING

- **Logical address Space (LAS) or Virtual Address Space (V.AS):**
  Represented in the form of words or bytes.
  Exp: 512 KW or 512KB.
- **Logical Address (LA) or Virtual Address (VA)**
  Represented in the form of bits.
  Exp: 32 bits.
- **Physical Address Space (PAS)**
  Represented in the form of words or bytes.
  Exp: 32 KW or 32KB.
- **Physical Address**
  Represented in the form of bits
  Exp: 16 bits

Ques. LA = 37 bits.
LAS = 2^{37} W
= 128 GW.

Ques. LAS = 512 MW
LA = ~~2^{29}~~ ~~26~~
= 29 bits

Ques. P.A = 46 bits.
PAS = 64 TB/64TW

Ques. PAS = 256GB
PA = 38 bits

WWW.INDIAPUSTAK.com 9560127002

- The technique of mapping CPU generated logical address to physical address is called as paging.

$\boxed{LA > P.A}$

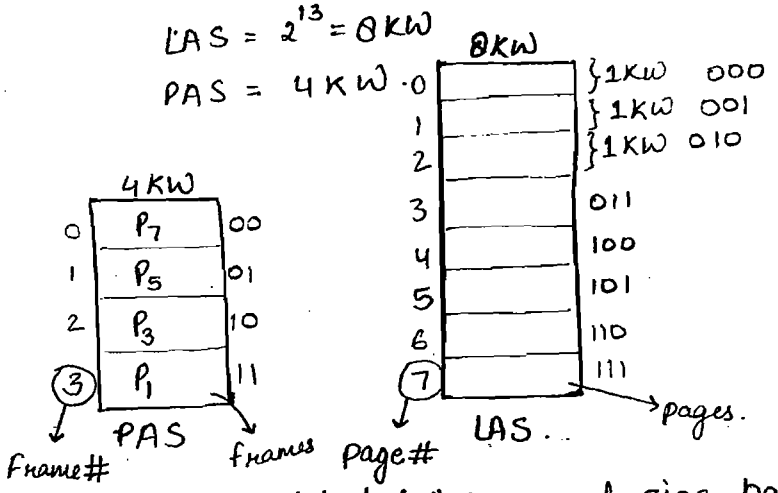
- Paging is implemented in the hardware level

Assume, LA = 13 bits

PA = 12 bits

$LAS = 2^{13} = 8KW$

PAS = 4KW



- LAS will be divided into equal size pages.

Assume, page size = 1KW.

$$\boxed{\#\text{ of pages} = \frac{LAS}{\text{page size}}} = \frac{8KW}{1KW} = 8$$

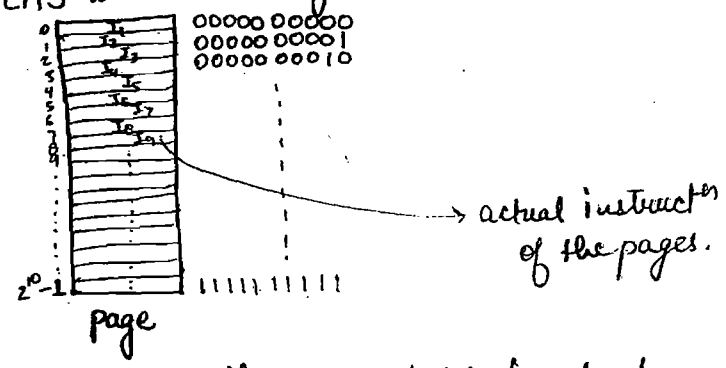
- PAS is divided into equal size frames.

$\boxed{\text{Page Size} = \text{Frame Size}}$

$$\boxed{\#\text{ of frames} = \frac{P.A.S}{\text{frame size}}} = \frac{4KW}{1KW} = 4.$$

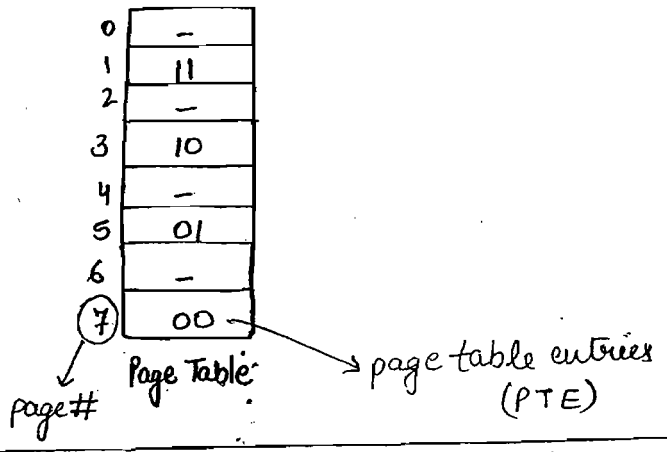
- Some of the pages of LAS will be brought into P.A.S.

Page Size = 1KW



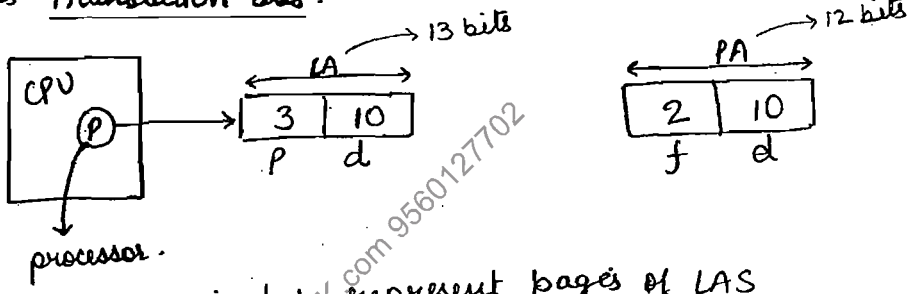
- whenever the paging is applied, the page table has to be maintained.
- No. of entries in the page table is same as No. of pages in the LAS.

WWW.INDIAPUSTAK.com 9560127772

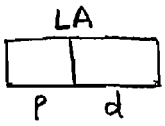


 # Entries in page table = # of pages in L.A.S

- **page table entry definately contains frame NO.**
- Page table is also called as <u>Address Translation Table</u> and Frame NO. is also called as <u>Translation bits</u>.

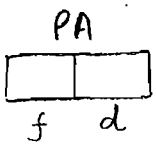


WWW.INDIAPICSTALK.com 9560127702



p = No. of bits required to represent pages of LAS
(OR)
Page Number.

d = No. of bits required to represent page size.
(OR).
Word Number of the Page.
(OR)
Page offset.



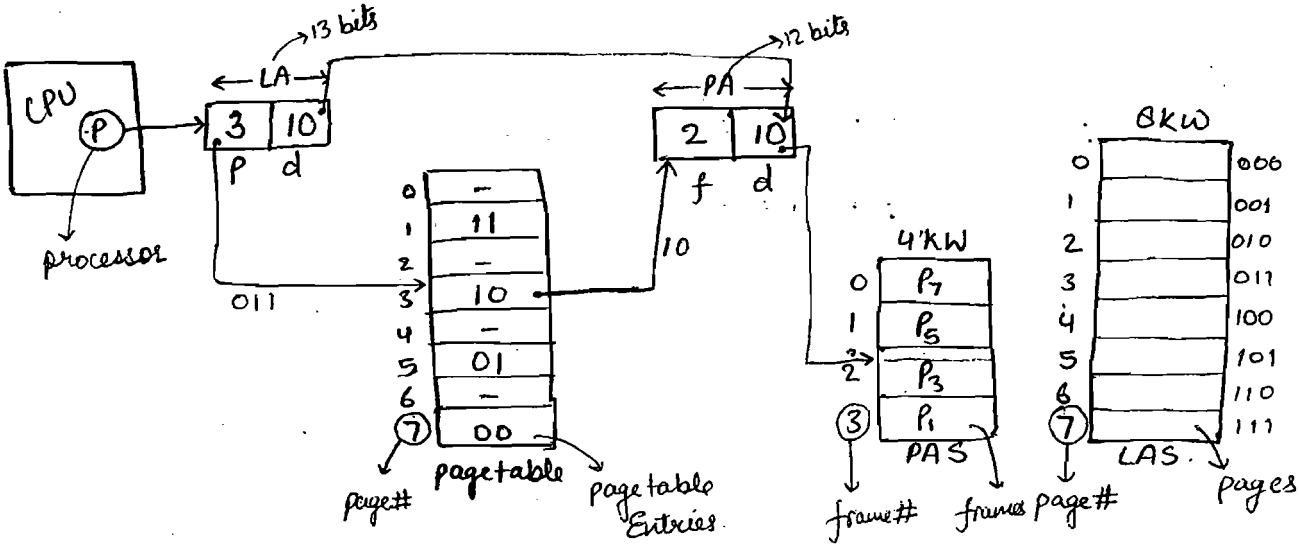
f = No. of bits required to represent frames of PAS.
(OR)
Frame Number.

d = No. of bits required to represent frame size
(OR)
Word Number of the Frame
(OR)
Frame offset.

Exp:

011	0000000101
p	d

10	0000000101
f	d



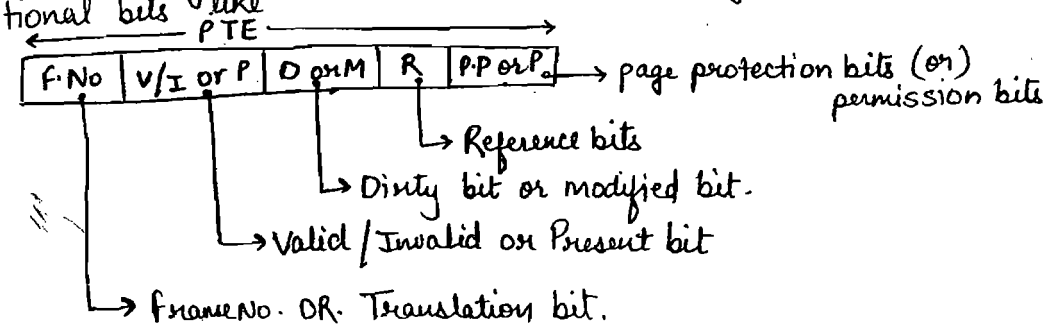
Important Points:

1. Whenever the process is created paging will be applied on the process. and Page Table will be created and the Base Address of the page Table will be stored in the P.C.B.
2. Paging is w.r.to every process and every process will have its own page table.
3. Page Tables of the processes will be stored in the Main Memory
4. There is No external fragmentation in the paging.
5. The internal fragmentation exists in the last page. and internal fragmentation in the paging is considered as $P/2$. where P = page size.
6. Maintaining the page table is considered as "overhead" for the system.

WWW.INDIAPUSTAK.com 9560127702

Note: Windows practically follows page size of 4KB.

Note: The page table entry definately contains frame No. but sometimes along with the frame No. it may also contain additional bits like



Valid / Invalid or Present bits

- This bit is used to identify whether the page is available or not available in the MM.

0 — page is not present in MM

1 — present.

Dirty Bit or Modified bit

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

0 — Not modified

1. — Modified.

Reference Bit

- This bit is used to identify how many times the page is referred by the processor.

Page Protection Bits OR Permission Bits

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

WWW.INDIAPUSTAK.com 9560027702

Ques. Consider a system whose LA = 27 bits and PA = 21 bits & page size = 4KW. and memory is word addressable. Then calculate ~~no.~~ no. of pages and no. of frames.

$$\#\text{ of pages} = \frac{LAS}{\text{page size}} = \frac{2^{27}W}{2^{12}W} = 2^{15} = \underline{32K}$$

$$\#\text{ of frames} = \frac{P.A.S}{\text{frame size}} = \frac{2^{21}W}{2^{12}W} = 2^{9} = \underline{512}$$

Ques. Consider a system which has no. of pages of 8K and page size is 16KB. and Memory is byte addressable. and physical Address is 22 bits. Then calculate logical address & no. of frames.

$$LAS = \#\text{ of pages} * \text{page size}$$
$$= 8K * 16KB$$
$$= 2^{13} * 2^{14} B = 2^{27} B.$$

$$\boxed{LA = 27 \text{ bits}}$$

$$\text{No. of frames} = \frac{PAS}{\text{frame size}} = \frac{2^{22}B}{2^{14}B} = 256.$$

Ques. Consider a system which has LAS = 128 MW and memory is Word Addressable. Physical Address is 24 bits. The PAS is divided into 8 K frames. Then what is the page size and how many pages in the L.A.S.

$$\text{Frame size} = \frac{PAS}{\#\text{ of frames}} = \frac{2^{24}\,W}{2^{13}} = \underline{2\,kW}.$$

$$\boxed{\text{frame size} = 2\,kW = \text{Page size}}$$

$$\#\text{ of pages} = \frac{LAS}{\text{Page size}} = \frac{128\,MW}{2\,kW} = \underline{64\,K}.$$

Ques. Consider a system with LA = 32 bits and PAS = 64 MB and Page size = 4KB. The memory is byte addressable. Then what is the approximate size of page table in Bytes.

a). 1 MB b). 2MB ✓ c). 4MB d) 8 MB.

$$\boxed{\text{P.T Size} = \#\text{ of Entries in P.T.} * \text{PTE Size}}$$

$$= \left(\frac{2^{32}}{2^{12}}\right) * \text{F.No. bits}$$

$$= 2^{20} * (14\text{ bits})$$

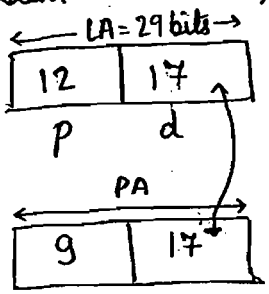
$$= 2^{20} * 2\,B$$

$$= \underline{2\,MB}.$$

$$\#\text{ of frames} = \frac{PAS}{\text{frame size}} = \frac{64\,MB}{4\,kB} = 2^{14}$$

WWW.INDIAPUSTAK.com 9560127702

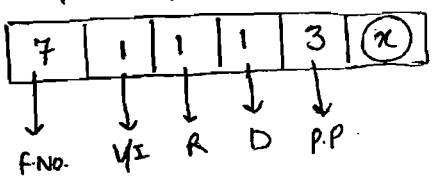
Ques. Consider a system having a page Table with the 4K entries and LA is 29 bits. Memory is word addressable then what is the ~~pages~~ PA if the system has 512 frames.



$9 + 17 = \underline{26\text{ bits}}.$

Ques. 40 (103).

PTE = 16 bits



$$\#\text{ of frames} = \frac{PAS}{\text{Frame Size}} = \frac{2^{16}\,B}{2^{9}\,B} = 2^{7}$$

$$16 - 13 = \underline{3}$$

Ques. Consider a system which has Logical Address = 40 bits and Page size = 16 KB and Memory is Byte Addressable. PTE size = 48 bits then. What is Page Table size 384 in MB?

PTsize = # of entries in PT * PTE size

$$= \left(\frac{2^{40}}{2^{14}}\right) * 48 \text{ bits}$$

$$= 2^{26} * 6 B$$

$$= 384 MB.$$

Ques. Consider a system which has LAS = PAS = 'S' Bytes. Page size = P Bytes. PTE size = 'e' bytes. Memory is Byte addressable. Then what is the optimal value of page size by minimizing the Memory overhead of maintaining PT size Internal Fragmentation in paging.

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

Memory overhead = PT size + IF in paging

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

$$\frac{1}{2} + \left(-\frac{s}{p^2} * e\right) = 0$$

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

$$\boxed{p = \sqrt{2se}.}$$

WWW.INDIAPUSTAK.com 9560127702

Performance of Paging:

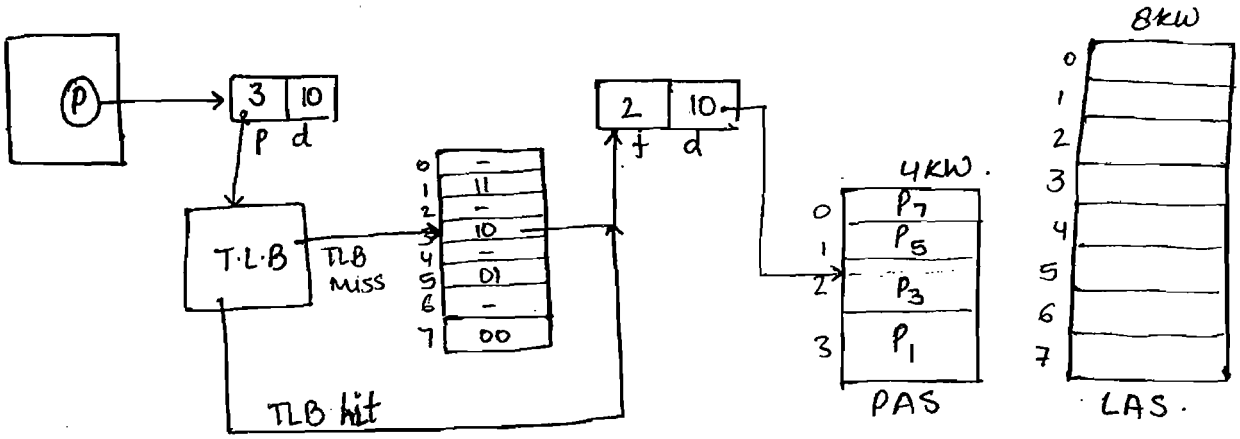
- The Main Memory access time = M.
- The page table are stored in MM then the formula for effective memory access time = 2M.

MMAT = M

E.M.A.T = 2M

- The translation look aside buffer (T.L.B) is added to improve the performance of paging.
- TLB is the hardware device implemented by using associative registers.
- TLB access time will be very less compared to MM access time.

- TLB contains frequently referred page Numbers & corresponding frame numbers.



- TLB access time = C
- TLB Hit Ratio = x

Then the formula for $\boxed{E.M.A.T = x(c+M) + (1-x)(c+2M)}$

Ques. Consider a system which has MM access time is 100 nsec. and TLB access time is 20nsec. and TLB hit ratio is 95%. Then what is the effective memory access time with TLB and without TLB.?

$$EMAT = x(m+c) + (1-x)[c+2m]$$
$$= 0.95(100+20) + 0.05[20+200]$$
$$= 125 \text{ nsec}.$$

Without TLB

$$EMAT = 2M = 2 \times 100 = 200\text{nsec}.$$

Ques. How much hit ratio is required to reduce the EMAT from 300ns without TLB to 250nsec with TLB. TLB access time = 60nsec.

$$2m = 300$$
$$m = 150$$

$$250 = x(60+150) + (1-x)(60+300)$$
$$250 = 210x + (1-x)(360)$$
$$250 = -150x + 360$$
$$150x = 110$$
$$x = \frac{11}{15}$$
$$x = 73\%$$

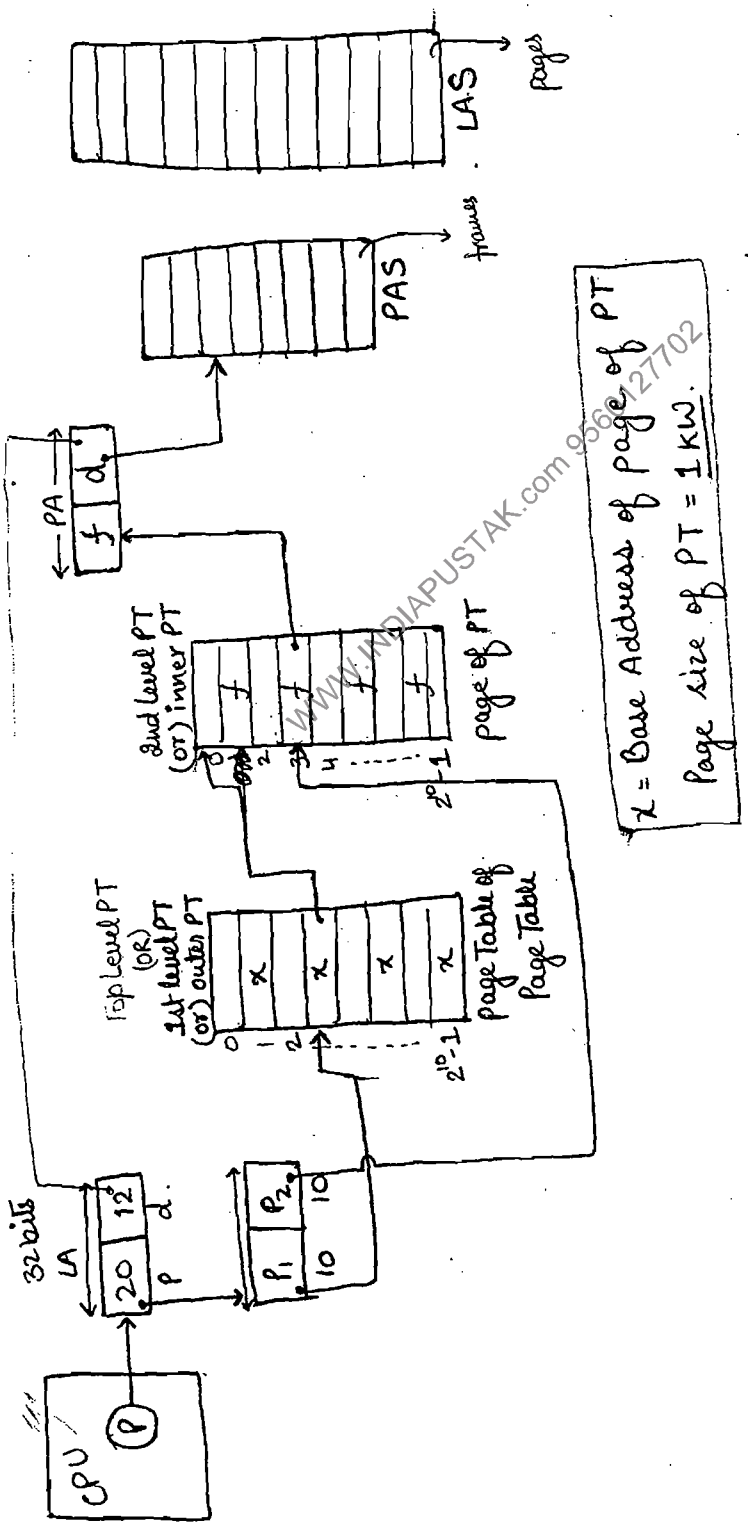
WWW.INDIAPUSTAK.com 9568127702

28/11/2019

Ques. Consider the system which has LA = 32 bits and Page size of LAS = 4KW and Page Table entry size = 1 word then what is page table size.

$$PT\ size = \frac{2^{32}}{2^{12}} * 1W = 2^{20}\ W = \underline{1MW}$$

Multi Level Paging:



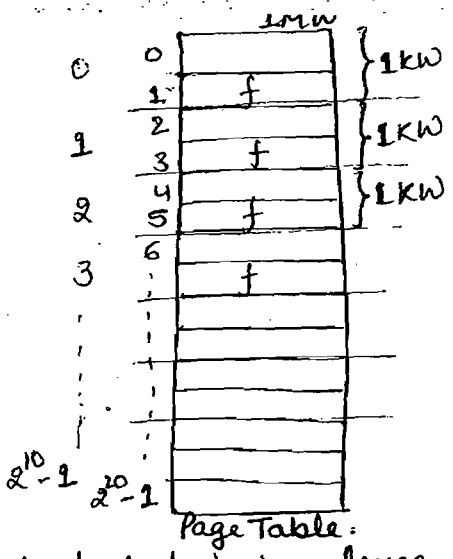
x = Base Address of page of PT

Page size of PT = 1KW.

P_1 = No. of bits required to represent pages of page Table.
(OR)
page No. of page table.

P_2 = No. of bits required to represent page size of page table
(OR)
word No. of page of page table.
(OR)
page offset of page table.

WWW.INDIAPUSTAK.com 9560127702



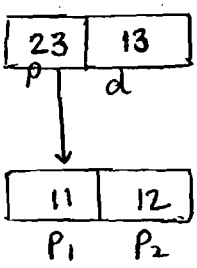
$$\boxed{\#\text{ of pages on P.T} = \frac{2^{20}\,W}{2^{10}\,W} = 2^{10} = \underline{1K}}$$

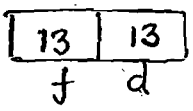
- To avoid the overhead of bringing large size Page Table into main memory. The concept of multi level paging is implemented.
- In the multilevel paging, paging will be applied on the page table and instead of bringing the entire page table into memory, the pages of P.T. will be brought into memory.

<u>Ques</u>: Consider a system which has 2 level paging applicable. The page table has divided into 2K pages and each page is having 4K entries. The ~~no~~ memory is word addressable. The PAS is 64M words, which is divided into 8K frames. Pagetable entry size in both the levels is 2W. Then calculate.

1. Length of logical address.
2. Length of physical address.
3. First level page table size.
4. Second level page table size (page of page table)

WWW.INDIAPOSTAK.COM 9509127702





$$\text{Frame size} = \frac{PAS}{\#\text{ frames}} = \frac{2^{26}}{2^{13}} = \underline{2^{13}\,W}$$

<u>Ans. 2</u> = <u>26 bits</u>

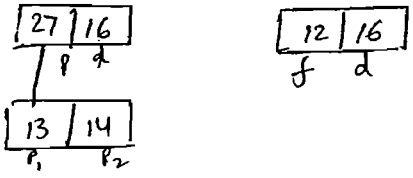
<u>Ans. 1</u> = 11+12+13 = <u>36 bits</u>

<u>Ans. 3</u> = $2^{P_1} * \text{PTE size} = 2^{11} * 2W = \underline{4KW}$

<u>Ans. 4</u> = $2^{P_2} * \text{PTE Size} = 2^{12} * 2W = \underline{8KW}$

Ques · Consider a system with the 2 level paging applicable. The PT has divided into 8K pages and each page is having 16K entries. Memory is byte-addressable. The PAS is 256 MB which is divided into 64KB frames. The Page Table Entry size in both the levels is 32 bits. Then calculate

1. Length of LA.
2. Length of PA
3. First level PT size.
4. Second level PT size (page of PT).



Ans.1 = 28 bits

Ans.2 = 43 bits

Ans.3 = $2^{13} * 4B = 32KB$

Ans.4 = $2^{14} * 4B = 64KB$.

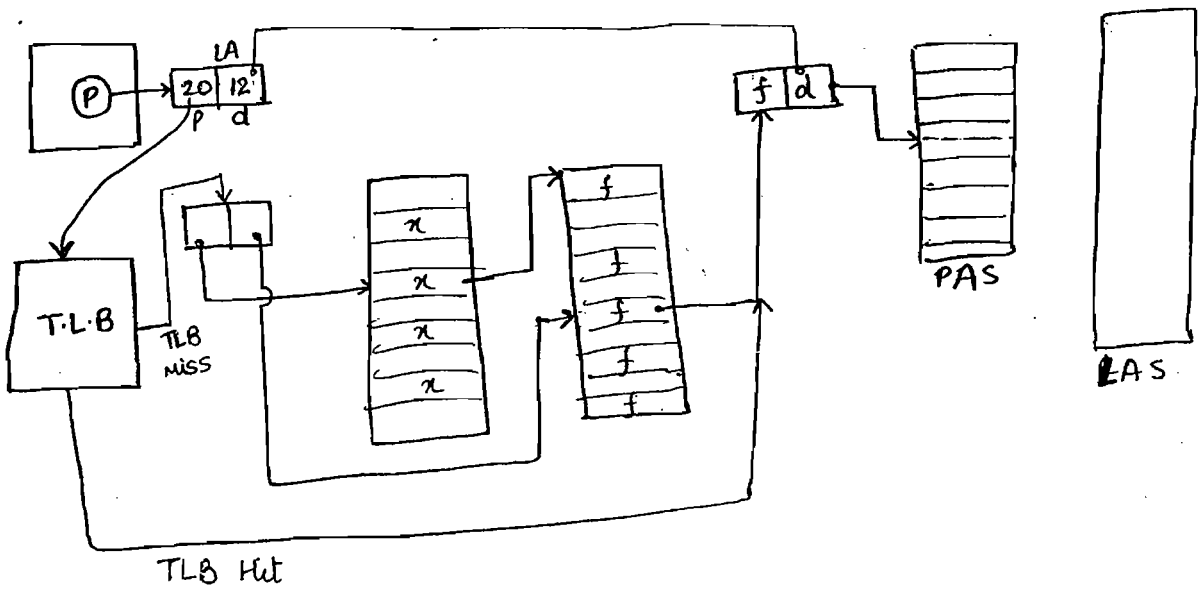
Performance of 2 Level Paging

- The main memory access Time M.M.A.T = m
- The page tables are stored in MM.
- Then the formula for Effective Memory Access Time.

$$\boxed{E.MAT = 3m}$$

- The TLB is added to improve the performance of paging.
- TLB contains frequently refered page numbers & corresponding frame numbers.

WWW.INDIAPOSTJOB.COM 9566127712



TLB access Time = 'c'

TLB Hit Ratio = 'x'

Then the formula for $\boxed{E.M.A.T = x(c+m) + (1-x)(c+3m)}$

⋮

for n level paging

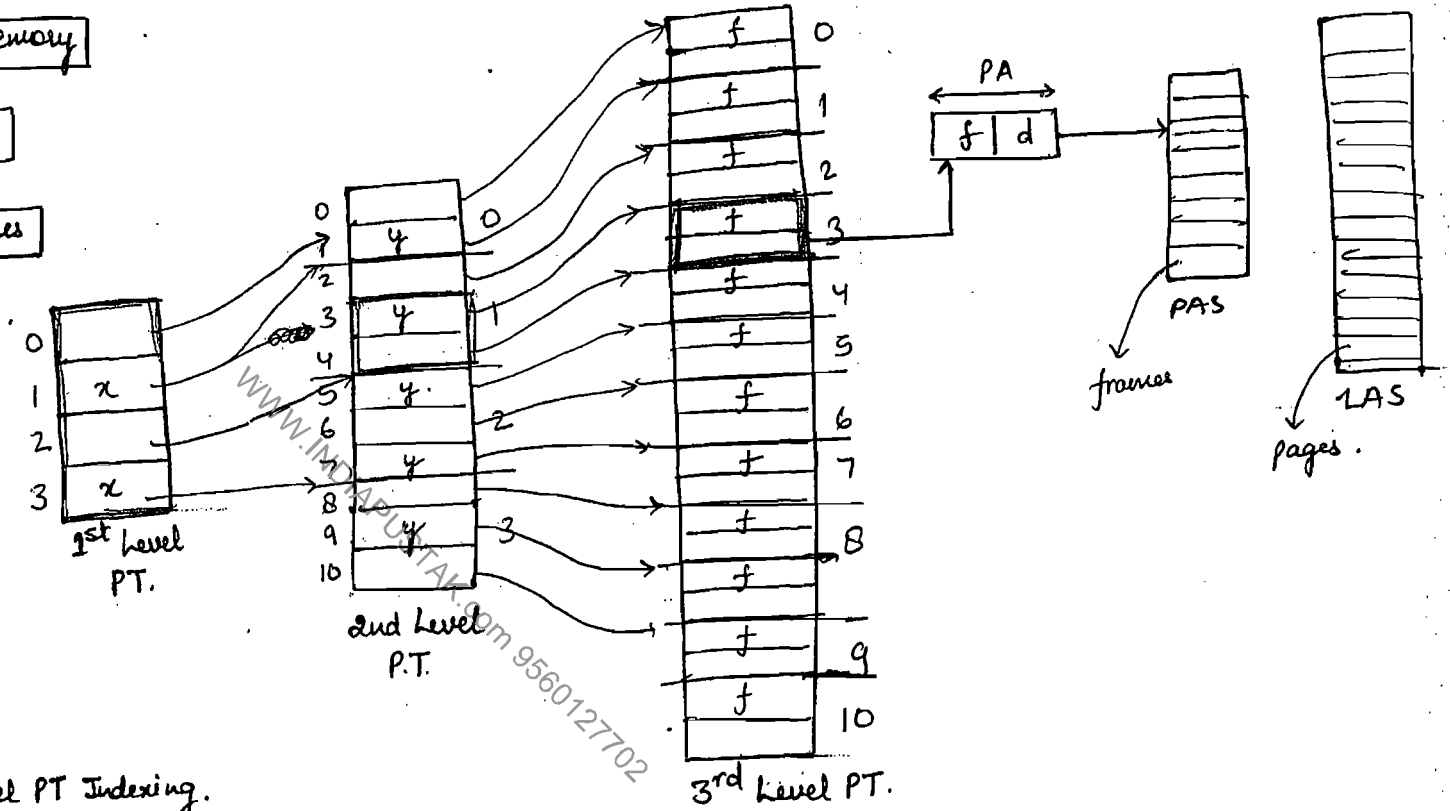
$$\boxed{E.M.A.T = x(c+m) + (1-x)(c + (n+1)*m)}$$

Important Points :

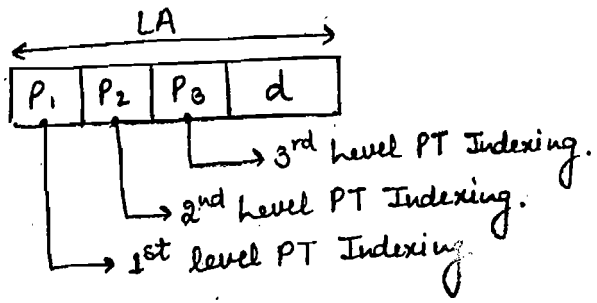
- In the multilevel paging when the paging applied on the page Tables, then the last page Table which we get is called as First Level Page Table.
- In the multilevel paging when the paging applied on the page tables then first level page table entries contain base address of the second level page table, the second level page table entries contain base address of the third level page table and so on... The final page table entry contains Frame Number of actual page.
- In the multi level paging when the paging applied on the page tables, then whatever may be the levels of paging, then all the page tables (Page of Page Table) will by stored in the Main Memory.
- In the multilevel paging when the paging applied on the page tables then whatever may be the levels of paging, all the page table entries contains frame No.
- If the page size is Not mentioned in the problem, then generally page size will be same in all the places. (Levels)

WWW.INDIANPUSTAK.com 9001012702

The PT's are stored in Main Memory
⇓
PAS
⇓
frames



# of bits in 'x'
=
# of bits in 'y'
=
# of bits in 'f'



WWW.INDIAPUSTAK.com 9560127702

⊛ Whenever the process is created paging will be applied on the process and Page Table will be created and the Base Address of the Page Table will be stored in the P.C.B.

Ques. 25 (102)

LA = 32 bits
Page Size = 1KB
PTE = 4B.

$\boxed{\text{PT Size} \leq \text{Page Size}}$

$\boxed{16\ MB \leq 1\ KB}$ ✗

$\boxed{64\ KB \leq 1\ KB}$ ✗

$\boxed{256\ B \leq 1\ KB}$ ✓

1st Time Paging

$$PT\ Size = \left(\frac{2^{32}}{2^{10}}\right) * 4B = 2^{22} * 2^{2} B = 2^{24} B = 16\ MB.$$

2nd Time Paging

$$PT\ Size = \frac{16\ MB}{2^{10} B} * 4B = 2^{16} B = 64\ KB$$

3rd Time Paging

$$PT\ size = \frac{64\ kB}{1\ kB} * 4B = 256\ B.$$

$\boxed{\text{1st level PT size}}$

Gate 2013

Ques. Consider a system which has LA = 46 bits & PA = 32 bits and memory is Byte Addressable. The PT entry size is 32 bits (4B) The OS uses 3 level paging for logical to PA Translation and first level PT size is exactly same as Page size. Then what is the Page size?

a) 2 kB b) 4 kB c) 8 kB ✓ d) 16 kB.

WWW.INDIANJUSTUK.com 9990127702

LA = 46 bits
PTE Size = 32 bits
Page Size = xB.

1st Time Paging

$$PT\ size = \frac{2^{46} B}{x\ B} * 4B = \frac{2^{48}}{x} B.$$

2nd Time Paging

$$PT\ Size = \left(\frac{2^{48}}{x} B \Big/ xB\right) * 4B = \frac{2^{50}}{x^2} B$$

3rd Time Paging

$$PT\ size = \frac{\frac{2^{50}}{x^2} B}{xB} * 4B = \frac{2^{52}}{x^3} B.$$

1st level PT size = Page size

$$\frac{2^{52}}{x^3} \cancel{B} = x \cancel{B}$$

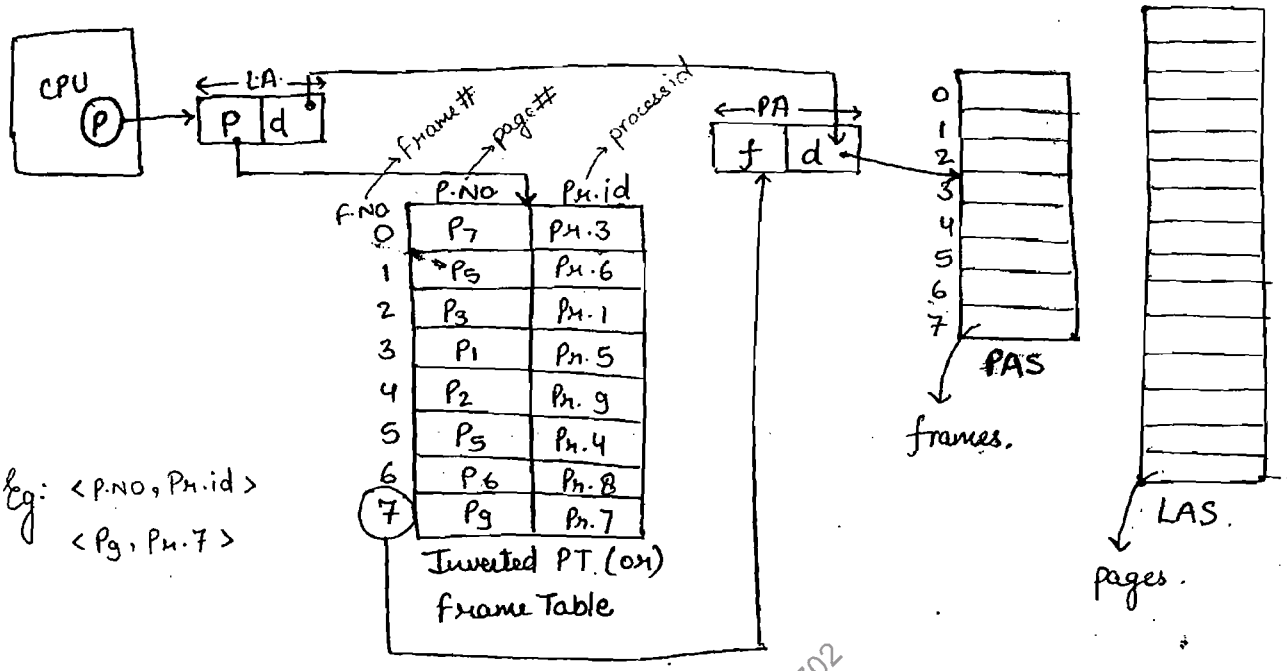
$$x^4 = 2^{52}$$

$$x^4 = (2^{13})^4$$

$$x = 2^{13}$$

$\therefore$ Page size $= 2^{13} B = 8kB$

INVERTED PAGING



Eg: <P.NO, Pr.id>
<P9, Pr.7>

- To avoid the overhead of maintaining P.T. for every process, the concept of inverted paging is implemented.
- In the inverted paging only one page table will be maintained for all the processes.

Note: No. of Entries in the Inverted PT is same as No. of frames in the PAS

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

Ques. Consider a system ~~LA~~ which has LA = 34 bits and PA = 29 bits & Page Size = 16 KB. PTE size is 8 Bytes then calculate
1. Conventional PT size.
2. Inverted PT Size.

Ans.1. PT size $= \frac{2^{34}}{2^{14}} * 8B = 2^{20} * 8B = \underline{8MB}$.

Ans.2. PT size = # of Entries in IPT * PTE size

$= \frac{2^{29}}{2^{14}} * 8B = 2^{18} B = \underline{256 KB}$.

WWW.INDIANPUSTAK.COM 9561127702

Ans. 5 (100)

IPT size = # of entries in IPT * PTE size

$= \frac{2^{30}}{2^{12}} * (\text{P.No} + 12 \text{ bits})$

$= 2^{18} * (20 \text{ bits} + 12 \text{ bits})$

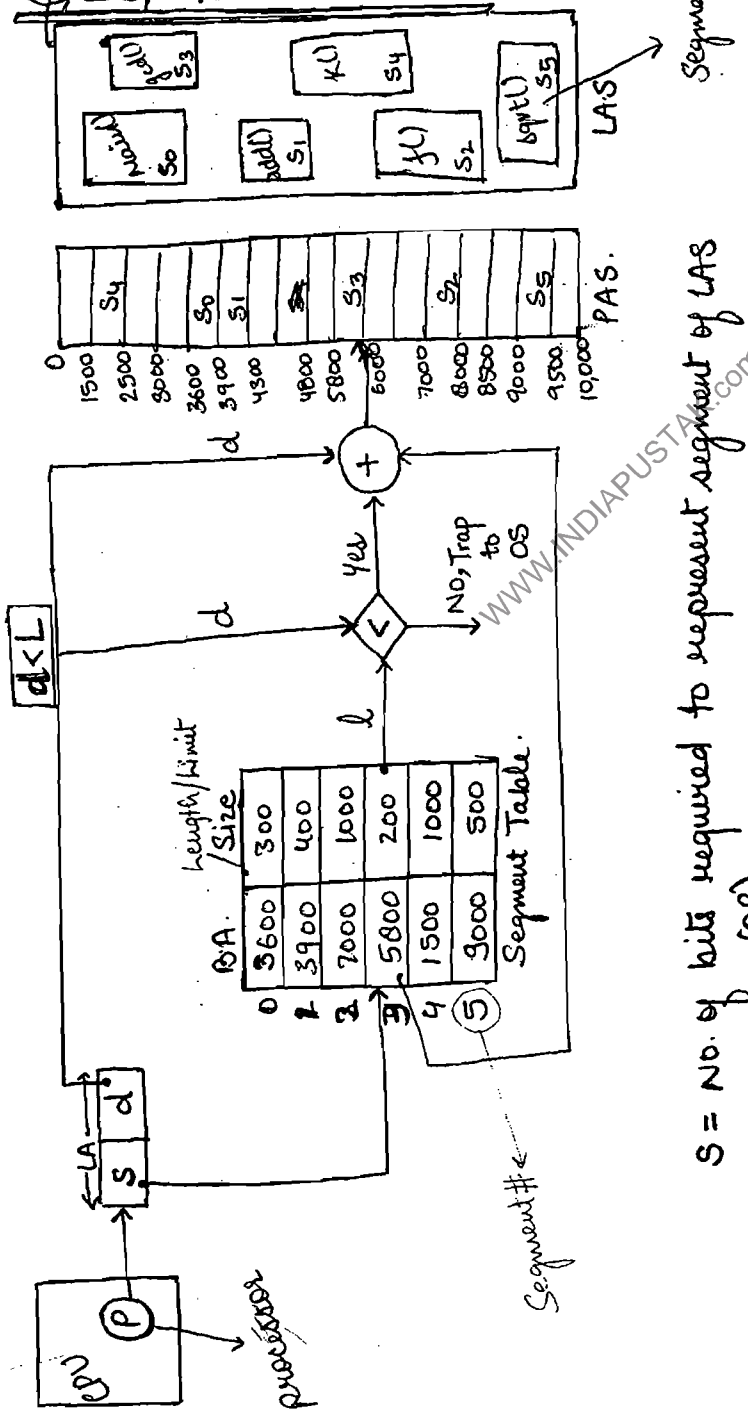
$= 2^{18} * 32 \text{ bits}$

$= 2^{18} * 4 \text{ B}$

$= 2^{20} \text{ B}$ (b)

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

SEGMENTATION


CPU
P
processor
LA
s
d
d<L
d
l
<
Yes
No, Trap to OS
+
d
Segment#
Segment Table
Length/Limit Size
BA
0 3600 300
1 3900 400
2 7000 1000
3 5800 200
4 1500 1000
5 9000 500
0
1500
2500
3000
3600
3900
4300
4800
5800
6000
7000
8000
8500
9000
9500
10,000
S4
S0
S1
S3
S2
S5
PAS
main() S0
read() S1
f() S2
log() S3
k() S4
length() S5
LAS
Segments

s = No. of bits required to represent segment of LAS
(OR)
Segment Number.

d = No. of bits required to represent segment size
(OR)
word number of segment
(OR)
Segment offset

WWW.INDIAPUSTAK.com 9560127702

- The paging does not follow users view of memory allocation. To achieve user's view of memory allocation the concept of segmentation is implemented.
- In the segmentation, LAS will be divided into various segments.
- The segments of LAS will vary in the size.
- Segments of LAS will be brought into PAS.
- Number of entries in the segment table is same as the number of segments in the LAS.

i.e. No. of entries in the segment Table = No. of segments in the LAS.

Exp: s = 3, d = 210.

d < L $\boxed{210 < 200}$ X

Trap to OS..

Exp: s = 3, d = 190.

d < L $\boxed{190 < 200}$ ✓

$\boxed{BA + d \Rightarrow AAS}$

5800 + 190 = 5990

- The variable size segments are brought from LAS to PAS, so it is similarly behaving like variable partition scheme. Hence segmentation still suffer from External Fragmentation.

WWW.INDIAPUSTAK.com 9560127702

29/11/2019

Segmented Paging

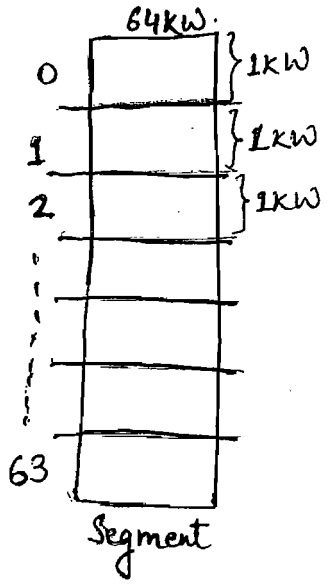
LA = 34 bits

<s, d> = (18, 16).

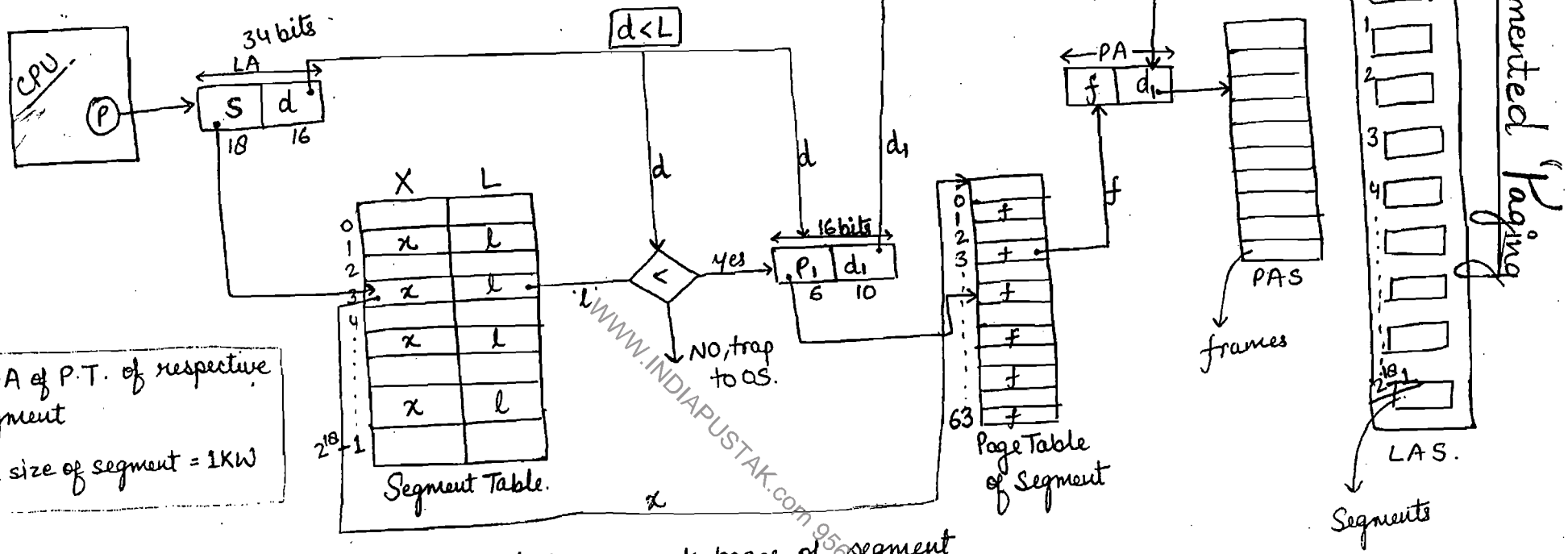
Segment Size = 2^{16} W = 64KW.

Page size of segment = 1KW.

$$\#\text{ of pages of segment} = \frac{\text{Segment Size}}{\text{Page Size}} = \frac{64KW}{1KW} = 64.$$



Paging on Segment ; when segment size Increases



x = B.A of P.T. of respective segment

Page size of segment = 1KW

P_1 = No. of bits required to represent pages of segment
(or)
page no. of segment.

d_1 = No. of bits required to represent page size of segment
(or)
word No. of page of segment.
(or)
page offset of segment.

WWW.INDIAPUSTAK.com 9560127702

- No of entries in the page table of segment is same as No. of pages on segment.

  i.e. $\boxed{\text{No. of entries in the PT of segment} = \text{No. of pages on segment}}$

- page size of segment is same as frame size of PAS.

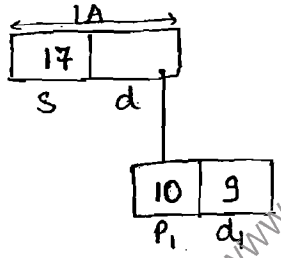
  i.e. $\boxed{\text{page size of segment} = \text{Frame size of PAS}}$

- No External Fragmentation is there.

Ques. Consider a system ~~si~~ using segmented paging architecture. The segment is divided into 1K pages and each page is ~~divi~~ having 512 entries. The segment No requires 17 bits to represent all the segment of LAS and frame No. requires 13 bits to represent all the frames of PAS. Memory is word addressable. Page Table entry size is 2W. Then calculate

1. length of LA 36
2. length of PA 22
3. PT size of segment 2KW

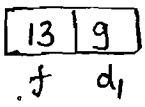
Ans. 1.



LA = 36 bits

Ans. 2

PA = 22 bits



Ans. 3. P.T size of segment = # of Entries in PT of segment * PTE size.

⇓

# of pages on Segment

⇓

1K * 2W

= 2KW

Ques. Consider a system using segmented paging architecture. The segment is divided into 8K pages and each page is having 2K entries. Memory is Byte Addressable. Segment No. requires 19 bits to represent all the segments of LAS. The PAS is 512 KB and page table entry size is 8 bits. Then calculate

1. length of LA.
2. length of PA.
3. PT size of segment.
4. No. of frames in PAS.

WWW.INDIAPUSTAK.com 9560127702

Ans.1 LA = 43 bits

19	
s	d

↓

13	11
P_1	d_1

Ans.2. PAS = 512KB.

PA = 19 bits.

Ans.3. PT size of segment = # of Entries in PT of segment * P.T.E size.

= 8K * 1B = 8KB.

Ans.4. # of frames

$\Rightarrow 2^t = 2^8$

$\Rightarrow 256.$

Ques 28: (102).

LAS = PAS = 2^{16}B.

LA = 16 bits

3	13
s	d

Segment Size = 2^{13} B.

Page size of segment = 2^x

PTE size = 2B.

PT Size of Segment = # of entries in PT of segment * PTE size.

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

$= 2^{13-x+1} B.$

$= 2^{14-x} B$

PT size of segment = frame size.

$2^{14-x} \cancel{B} = 2^x \cancel{B}$

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

$x = 7$

Page size of Segment = $2^x B = 2^7 B$

= 128 B.

WWW.INDIANSTAK.com 9560127702

5 VIRTUAL MEMORY

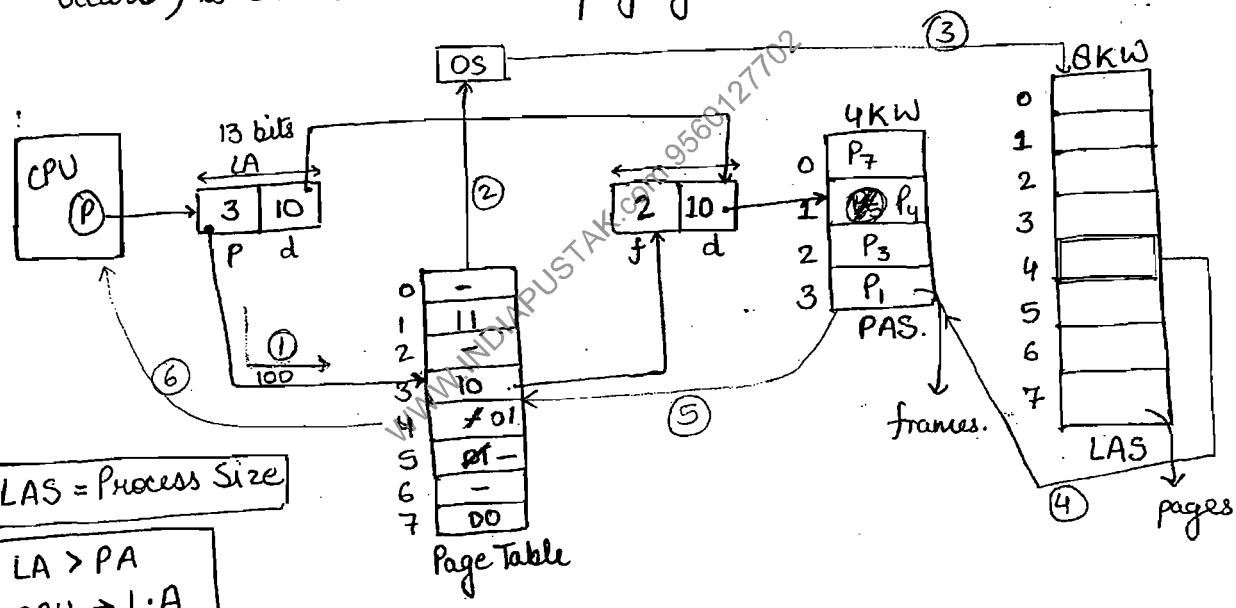
Defination: Virtual Memory gives an illusion, to the programmer, that the programs of larger size than actual physical memory can be executed.

Virtual Memory is implemented by

- demand paging
- demand segmentation

Demand Paging

- Loading the page into memory on demand. (whenever page fault occurs) is called as demand paging.



- LAS = Process Size
- LA > PA
  CPU → L.A

Step 1: CPU is trying to refer the page in MM, but the page is currently Not available in the MM. (P.A.S) (This is called as "Page Fault")

Step 2: The program execution will stop & signal will be sent to the OS. regarding the page fault.

Step 3: The OS will search for the required page in LAS. (virtual memory).

Step 4: The required page is brought from LAS to PAS.
Page Replacement Algorithms are used for the decision making of selecting the page for the replacement.

WWW.INDIAPUSTAK.COM 9560127702

Step.5: The respective P.T.E will be updated accordingly.

Step.6: The signal will be sent to the CPU to continue the program execution. & the CPU will access the required page in the MM. and continue program execution.

Note • The main purpose of the virtual memory is by allocating less memory to the process, we are trying to execute large size process & more no. of processes. Indirectly, we are trying to increase degree of multiprogramming.

• If the Virtual Memory is removed from the system then efficient implementation of multiprogramming & multiuser Not possible. (System performance will Slow Down).

Page Fault Service Time (P.F.S.T)

• The time taken to service the page fault is called as page fault service time.

- P.F.S.T = 's' (all 6 steps included).
- MM access Time = m
- Page fault rate = p

Then the formula for E.M.A.T

$$\boxed{EMAT = p*s + (1-p)*m}$$

actual it is 2m but Page Table access time is neglected.

$$\boxed{s >> m}$$

s → ms, m → ns

www.INDIANUSTAK.com 9560127702

Ques. Consider a system which has MMAT of 35ns and Page Fault Service Time is 175ns and Page Hit Ratio is 75% then what is Effective Memory Access Time?

$$E.MAT = p*s + (1-p)*m$$

$$= 0.25 * 175 + 0.75 * 35$$

$$= 43.75 + 26.25$$

$$= \underline{70\ ns}.$$

Shaun Nagar
Extention
Dr Ms

Ques. 20 (107)

$m = i$

$S = (i+j)$

$p = \frac{1}{K}$

$$EMAT = p * S + (1-p) * m.$$

$$= \frac{1}{K}(i+j) + \left(1-\frac{1}{K}\right) * i.$$

$$= \underline{i + \frac{j}{K}}$$

Ques (37) (109)

a). 35 ns
~~b)~~ 30 ns
c) 27 ns
d). 25 ns.

$m = 20ns.$

$p = \frac{1}{10^6}$

$S = 10ms + 20ns.$

$$EMAT = \frac{1}{10^6}\left(10 * 10^{-3} + 20 * 10^{-9}\right) + \left(1 - \frac{1}{10^6}\right) * 20ns$$

$$= 10ns + 20ns = \underline{30ns}.$$

Ques (13) (106).

$$EMAT = \underline{p * S} + (1-p) * m.$$

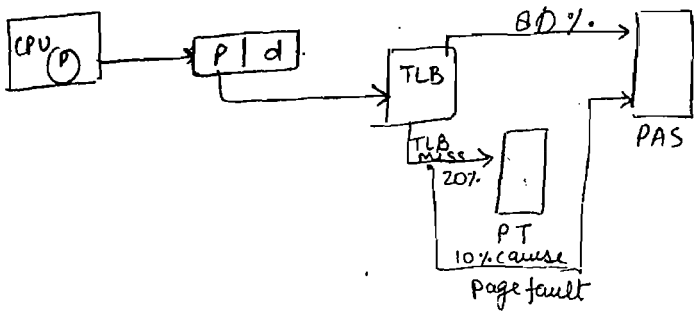
Modified
70%
20ms

not modified
30%
8ms.

$$2ms = p(0.7 * 20ms + 0.3 * 8ms) + (1-p) * 1ms$$

$$\boxed{p = 0.064}$$ (b) ✓

Ques (21) (107).



$$EMAT = 0.80(0 + 10ms) + 0.20(0 + 0.10 * 200ms + 0.90 * 10ms) = 13.8$$

TLB Hit Ratio | TLB Time | MMAT | TLB Miss | TLB Time | PF Rate | PFST | No PF | MMAT

WWW.INDIAPUSTAK.com 9560127702

Ques. The amount of memory on a system is inversely proportional to page fault rate. Each time when memory doubles, then the page fault rate is reduced by third (p is made as $p/3$). Currently system has 32 MB of memory and page fault rate is 2%. The MM access time is 500ns and EMAT = 300 μs. If the memory is increased to 128 MB. then what would be the EMAT ____ in μs ?

$$EMAT = p * s + (1-p) * m$$

$$300 * 10^3 ns = 0.02 * s + 0.98 * 500 ns.$$

$$s = 14975500 \text{ nsec.}$$

$$EMAT = \frac{0.02}{9} * 14975500 \text{ ns} * \frac{8.98}{9} * 500 \text{ ns}$$

$$= 33278.888 + 498.88$$

$$= 33777.77 \text{ ns.}$$

$$= \underline{33.77 \ \mu sec}.$$

Mem.	P.F rate	EMAT
32MB	0.02	300μsec.
↓	↓	↓
64MB	$\frac{0.02}{3}$	
↓	↓	
128MB	$\frac{0.02}{9}$	

WWW.INDIANJUSTAK.com 9560127702

PAGE Replacement Algorithms:

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

↳ page number referred by the process in LA.

Note: No. of frames allocated to the process is decided by instruction set architecture.

① First In First Out (FIFO):

4 Frames.

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

Total PF = 10.

3 Frames.

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

Total PF = 15

Ques. Reference String: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.
Use FIFO.

4 Frames.

1	2	3	4	1	2	5	1	2	3	4	5
			4	4	4	4	4	4	3	3	3
		3	3	3	3	3	3	2	2	2	2
	2	2	2	2	2	2	1	1	1	1	5
1	1	1	1	1	1	5	5	5	5	4	4
F	F	F	F			F	F	F	F	F	F

Total PF = 10.

3 Frames

1	2	3	4	1	2	5	1	2	3	4	5
		3	3	3	2	2	2	2	2	4	4
	2	2	2	1	1	1	1	1	3	3	3
1	1	1	4	4	4	5	5	5	5	5	5
F	F	F	F	F	F	F			F	F	

Total PF = 9.

WWW.INDIABUSTAS.COM 9560127702

BELADY'S ANOMALY:
By increasing No. of frames of the process, the no. of page fault should decreasing. But instead they increasing. This problem is called as 'Beleady's Anoma

(2). Optimal Page Replacement:

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

4 Frames

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

Total PF = 8.

3 frames

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	~~1~~	3	3	3	3	3	3	3	~~3~~	1	1	1	1	1	1	1
	0	0	0	0	0	~~0~~	4	4	~~4~~	0	0	0	0	0	0	0	0	0	0
7	7	~~7~~	2	2	2	2	2	2	2	2	2	2	2	2	2	~~2~~	7	7	7
F	F	F	F		F		F			F			F				F		

Total PF = 9.

30/11/2019

3 LRU Page Replacement :

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

3 frames

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	~~1~~	3	3	~~3~~	2	2	2	2	2	2	2	2	~~2~~	7	7	7
	0	0	0	0	0	0	0	~~0~~	3	3	3	3	3	~~3~~	0	0	0	0	0
7	7	~~7~~	2	2	2	~~2~~	4	4	~~4~~	0	0	~~0~~	1	1	1	1	1	1	1
F	F	F	F		F		F	F	F	F			F		F		F		

Total PF = 12

www.INDIAMUSTAK.com 9560127702

4 frames

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			2	2	2	2	2	2	2	2	~~2~~	2	2	2	2	2	2	2	~~2~~
		1	1	1	1	~~1~~	4	4	4	4	4	~~4~~	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	7	~~7~~	3	3	3	3	3	3	3	3	3	3	3	~~3~~	7	7	7
F	F	F	F		F		F						F				F		

Total PF = 4

4 MRU Page Replacement :

In the event of page fault, replace the page which is most recently used.

4 frames

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			2	2	2	2	2	~~2~~	~~3~~	~~0~~	~~3~~	2	2	~~2~~	0	0	0	0	0
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	~~0~~	~~3~~	~~0~~	4	4	4	4	4	4	4	4	4	4	4	4	4
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
F	F	F	F		F	F	F		F	F	F	F			F				

Total PF = 12.

3 Frames

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

Total PF = 16.

Ques 17. (107) Page Size = 100 records

Record #	Page #
0-99	0
100-199	1
200-299	2
300-399	3
400-499	4
500-599	5.

Addresses ⇒ 100, 200, 400, 499, 510, 530, 560, 120, 220, 240, 260, 320, 370.

Reference String ⇒ 1, 2, 4, 5, 1, 2, 3.

∴ 7

Note • The reference String will never contain continuous same page.

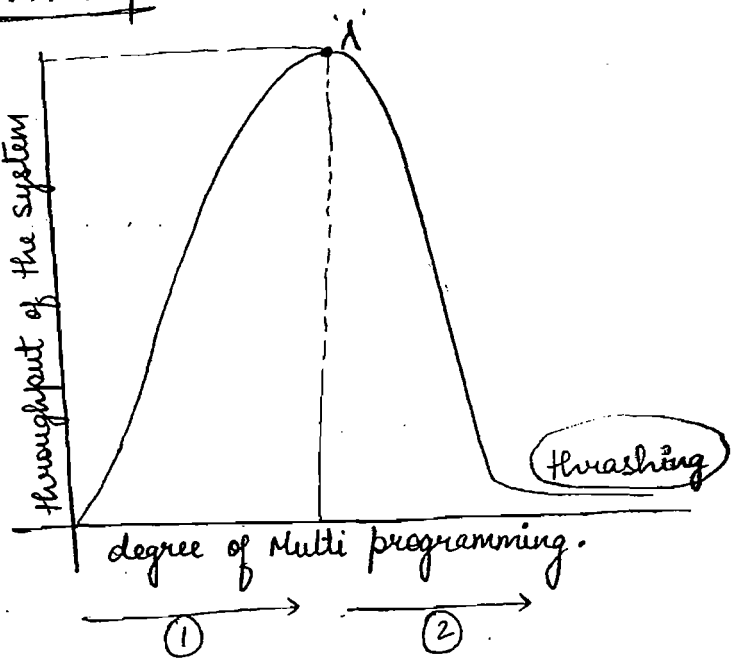
Eg: 1, 2, 3, 4, 4, 3, 2, 1 ✗

1, 2, 3, 4, 3, 2, 1 ✓

• If there is only one frame allocated to the process, then every page referred will be a page fault.

WWW.INDIAPUSTAK.com 9560127702

THRASHING



Exp: Free frames = 400

① # of processes = 100.
Each process P_i gets ⇒ 4 frames.

② # of processes ⇒ 400
Each process P_i gets ⇒ 1 frames

- In the initial degree of multiprogramming, upto the point of 'δ' the CPU utilization is very high and throughput of the system is also very high and the system resorces are utilized perfectly.
- If we further increase degree of multiprogramming, the throughput of the system will drastically fall down. and system will spend more time only in the page replacement and time taken to complete execution of the process increase. This situation in the system is called as Thrashing.

WWW.INDIAPHSTAK.com 9560127702

Causes of Thrashing:

1. High degree of Multiprogramming.
2. Lack of frames. (PAS).

Recovery of Thrashing:

1. Do not allow the system to go into thrashing & instruct the long term scheduler not to bring the processes into the memory after the point of δ.
2. If the system is already in the thrashing then instruct the Mid Term Scheduler to suspend some of the process, so that we can recover the system from thrashing.

Ques. Consider a system with the following parameters.

CPU Utilization = 10 %
Paging Disk = 90% → Time spend on page replacement

} Thrashing

Which of the following factors will improve the CPU utilization

1. Install faster CPU.
2. Install Bigger disk
3. Increase degree of Multiprogramming.
4. ~~Install more Main memory~~

4. Decrease degree of Multiprogramming.
5. Install More MM.
6. Decrease page Size
7. Increase page Size

Ans. System is in Trashing.

1. No, because problem is in memory.
2. No, because problem is in MM.
3. No.
4. Yes.
5. Yes.
6. NO
7. LIKELY., because less removement in the CPU utilization.

WWW.INDIAPUSTAK.com 9560127702

Chapter-4

File Systems

File : Collection of logically related entities (records).

Attributes :

1. Name
2. Type
3. Size
4. Location.
5. Permission
6. Creation Date
7. Last Modified Date
8. Owner
9. Password.

} File Context ⇓ FCB (File Control Block)

- All the attributes of the file is known as File Context
- File Context will be stored in the F.C.B (**File Control Block**).

WWW.INDIANJUSTAK.com 9560127002

Operations : Various operations performed on the file.

1. create
2. open
3. write
4. read
5. edit
6. save
7. save as
8. Rename
9. hide.
10. copy
11. paste
12. cut
13. delete
14. send
15. print
16. Redo/Undo.

Types : Various types of file are:

1. .doc
2. .txt
3. .exe
4. .obj
5. .class
6. .bat
7. .dll
8. .png.
9. .pdf
10. .jpg
11. .xls / .xlsx
12. .mp3 / .mp4
13. .avi / .FLV / .mkv / .3gp.
14. .c / .cpp / .java / .xml / .php / .html / .javascript / .css / .apk
15. .rar
16. .jar

⋮

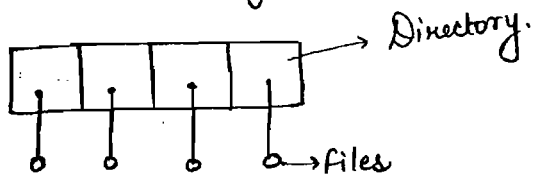
Access Methods: File is having various access methods.

1. Sequential access:
2. Random Access.

Note: For the better classification of the files, files will be stored in the directory.

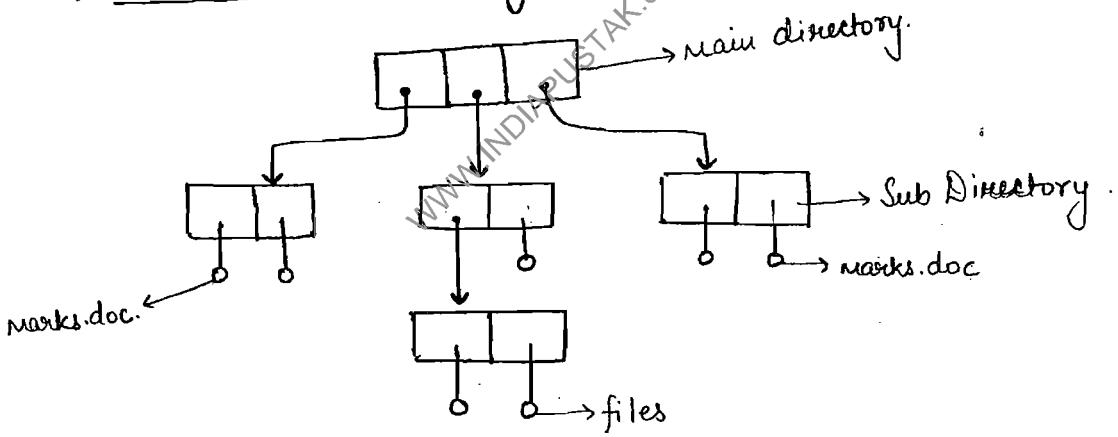
DIRECTORY STRUCTURE:

1). Single Level Directory



- Implementation of the directory structure easy or simple.
- Two files cannot have the same Name.
- Searching time for the files will be more.

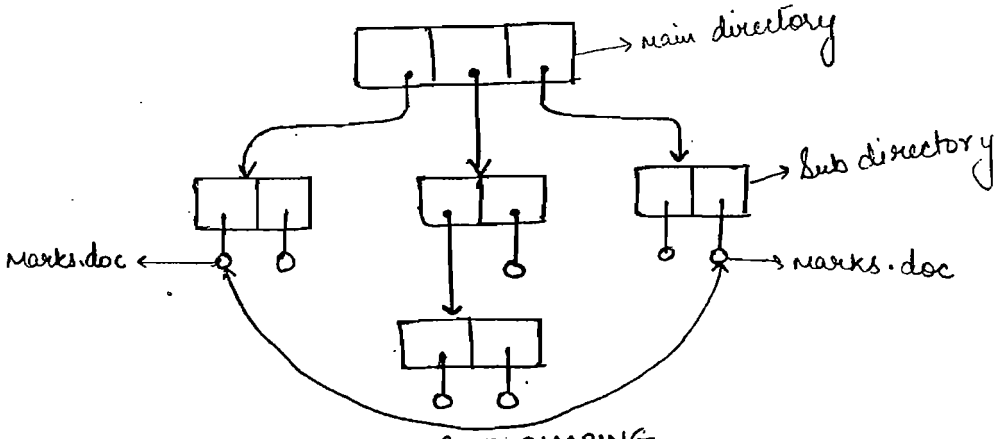
2). Multi Level Directory Structure



- Implementation of the directory structure is difficult or complicated.
- Better classification of the files as per the criteria.
- Searching Time for the files will be less.
- If the same file exists in 2 different directories, then if one file is updated, then other ~~w~~ file has to be updated accordingly otherwise, there will be a inconsistency.

WWW.INDIAPUSTAK.COM 9560127702

3). Acyclic Graph Directory Structure

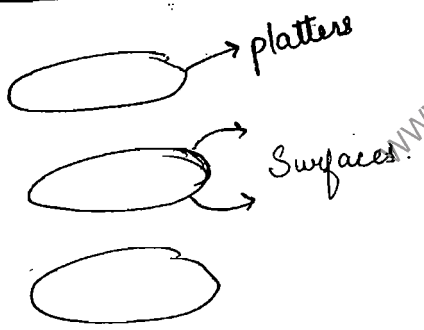


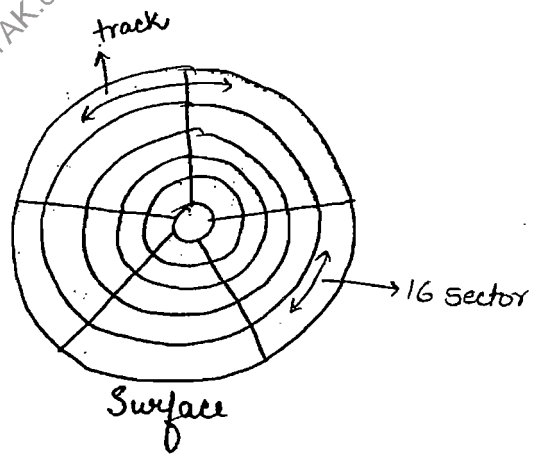
FILE SHARING.

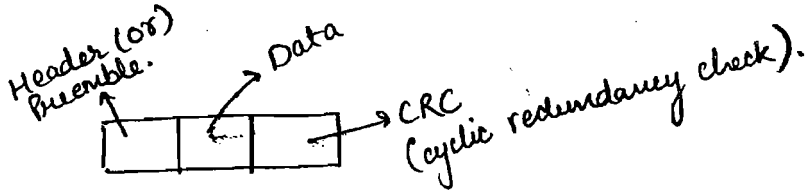
- Implementation of directory structure is difficult.
- The better classification of the files as per the criteria.
- Searching time for the files will be less.
- If the same file exist in the two different directories then if one file is updated then the other file will be updated automatically. by using File Sharing Concept.

WWW.INDIAPUSTAK.com 9560127702

DISK STRUCTURE :







Sector (or)
⊛Disk Block.

Ques. Consider a disk which has 16 platters and each platter is having 2 surfaces. Each surface is divided into 1K tracks and each track is further divided into 512 sectors and each sector can store 2KB data. Then calculate.

1. Capacity of the disk.
2. How many bits are required to identify, the specific sector of the disk ?

Ans.1. $2^4 * 2 * 2^{10} * 2^9 * 2^{11} B$

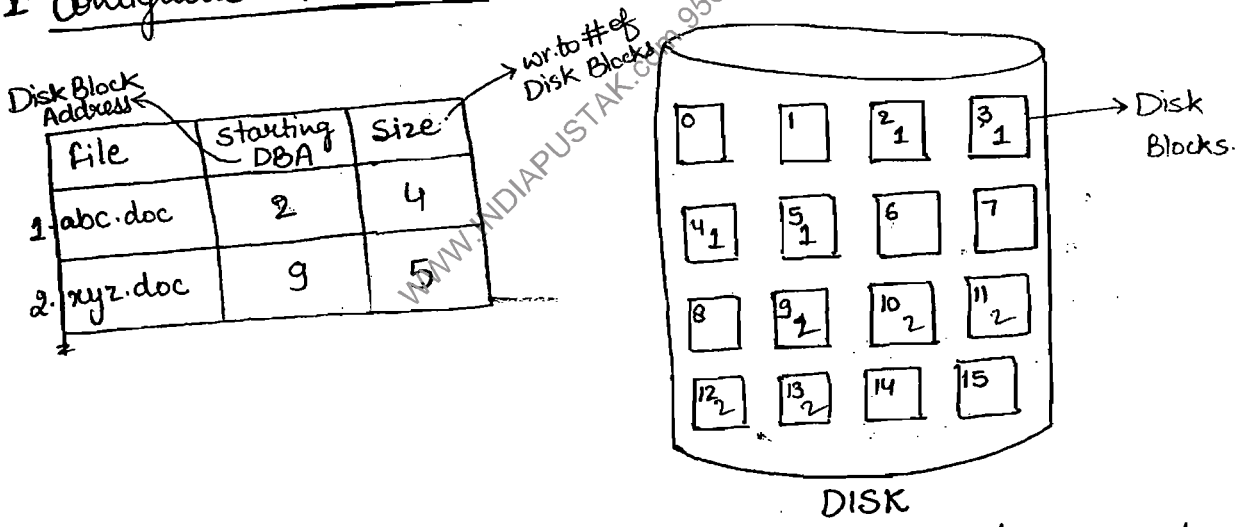
$\Rightarrow 2^{35} B$

$\Rightarrow$ 32GB..

Ans.2. $4+1+10+9 = 24$ bits..

DISK SPACE ALLOCATION Methods

1. Contiguous Allocation :



	File	Starting DBA	Size
1	abc.doc	2	4
2.	xyz.doc	9	5

WWW.INDIAPUSTAK.COM 9560127702

- In the continuous allocation, whenever the file is created, disk blocks are allocated in a continuous manner and
- every file is associated with the two parameters.
  1. Starting DBA
  2. size
- Increasing the file size may not be possible always.
- It is suffering from External Fragmentation.
- Internal fragmentation may exist in the last disk block of the file.

- It supports both Sequential and Random access of the file.

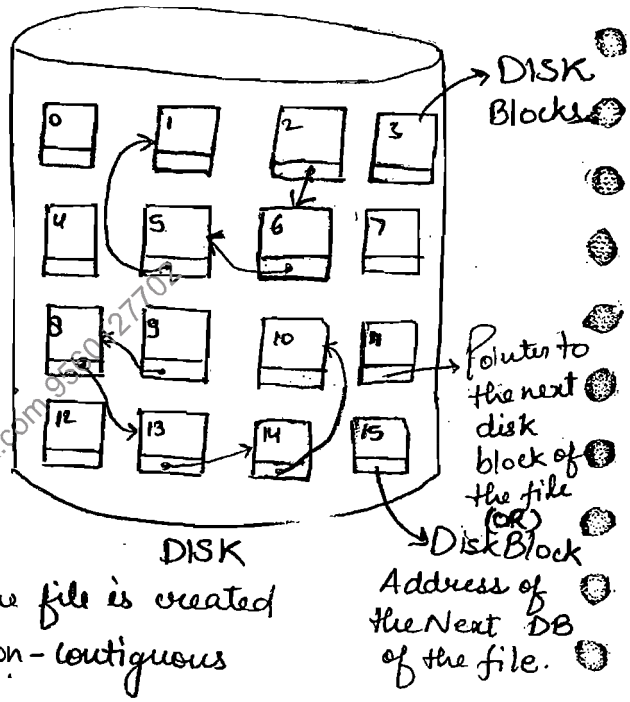
$$\boxed{\text{Starting DBA} + \text{Offset} \Rightarrow \text{Random Access}}$$

eg: for going on the fifth disk block for xyz.doc. then

$$9 + 4 = \underline{13}.$$

2. Linked Allocation (non-Contiguous Allocation)

File	Starting DBA	Ending DBA
abc.doc	2	1
xyz.doc	9	10



WWW.INDIAPUSTAK.com 9990227702

- In the linked allocation, whenever the file is created the disk block are created in the Non-contiguous manner.
- Every file is associated with 2 parameters:
  1. Starting DBA
  2. Ending DBA.
- Increasing the file size is always possible if free disk blocks is available.
- There is no external fragmentation.
- Internal Fragmentation may exist in the last disk block of the file.
- There is a overhead in maintaining a pointer in every disk block.
- If the pointer of any disk block is lost, file will be truncated.
- It supports only sequential access of the file.

Ques. Consider a disk which has 16 platters and each platter is having 2 surfaces. Each surface is divided into 1K tracks and each track is further divided into 512 sectors and each sector can store 2KB data. Then calculate.

1. Capacity of the disk.
2. How many bits are required to identify, the specific sector of the disk ?

Ans.1. $2^4 * 2 * 2^{10} * 2^9 * 2^{11} B$

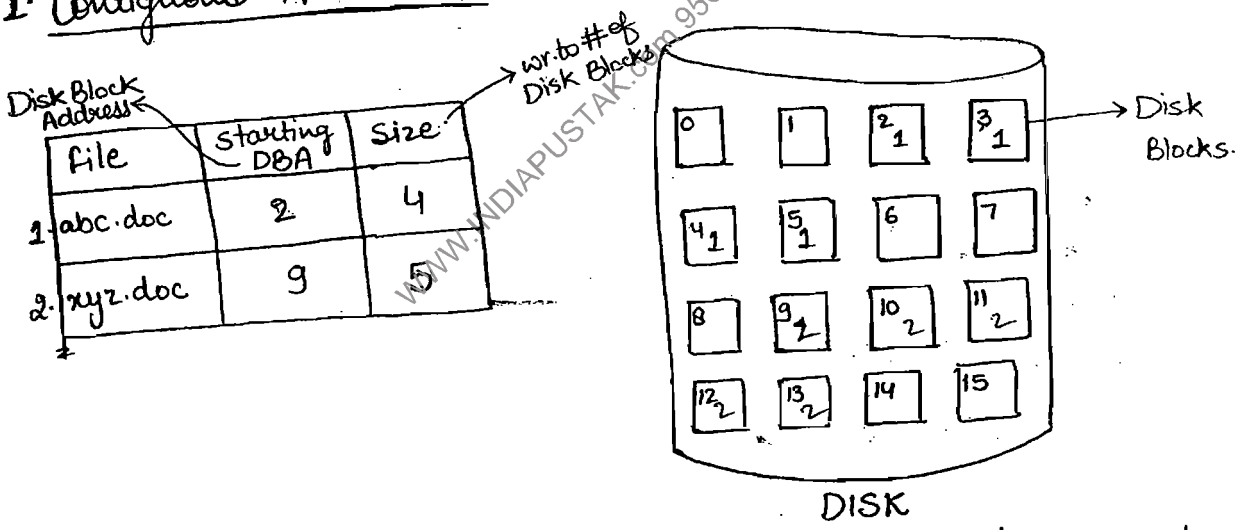
$\Rightarrow 2^{35} B$

$\Rightarrow$ 32GB..

Ans.2. $4+1+10+9 = 24$ bits..

DISK SPACE ALLOCATION Methods

1. Contiguous Allocation :



	File	Starting DBA	Size
1.	abc.doc	2	4
2.	xyz.doc	9	5

WWW.INDIAPUSTAK.COM 9560127702

- In the continuous allocation, whenever the file is created, disk blocks are allocated in a continuous manner and
- every file is associated with the two parameters.
  1. Starting DBA
  2. Size
- Increasing the file size may not be possible always.
- It is suffering from External Fragmentation.
- Internal fragmentation may exist in the last disk block of the file.

- It supports both Sequential and Random access of the file.

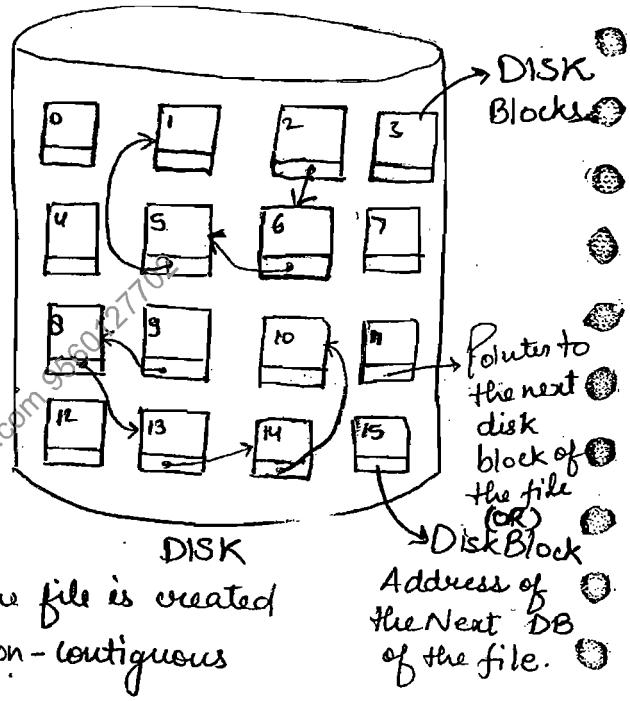
$$\boxed{\text{Starting DBA} + \text{Offset} \Rightarrow \text{Random Access}}$$

eg: for going on the fifth disk block for xyz.doc. then

$$9 + 4 = \underline{13}.$$

2. Linked Allocation (non-Contiguous Allocation)

File	Starting DBA	Ending DBA
abc.doc	2	1
xyz.doc	9	10

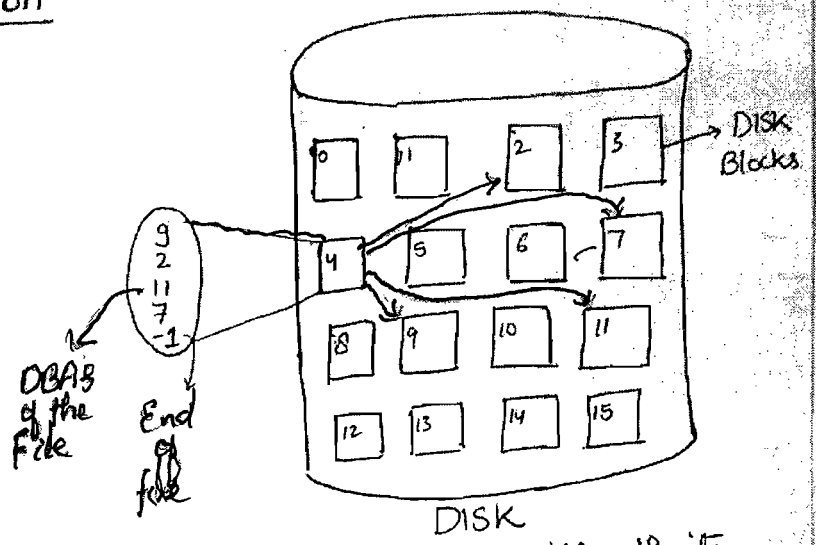


WWW.INDIAPUSTAK.com 9560127702

- In the linked allocation, whenever the file is created the disk block are created in the Non-contiguous manner.
- Every file is associated with 2 parameters:
  1. Starting DBA
  2. Ending DBA.
- Increasing the file size is always possible if free disk blocks is ~~f~~ available.
- There is No external fragmentation.
- Internal Fragmentation May exist in the last disk block of the file.
- There is a overhead in maintaining a pointer in every disk block.
- If the pointer of any disk block is lost, file will be truncated.
- It supports only sequential access of the file.

3. Indexed Allocation

File	IndexNode
abc.doc	4



- In the indexed allocation, every file is associated with its own index node.
- Index node contains all the disk block addresses of the file.
- If the file is very-very large then one disk block may not be sufficient to store all the disk block addresses.
- If the file is very very small, then it is waste of using entire one disk block just to store the addresses.
- There is no external fragmentation but internal fragmentation may exist in the last disk block of the file.
- It supports both sequential and random access of the file.

WWW.INDIAPOLISTAN.com 9560327...

Imp

4. Unix I-Node Implementation

(An Extension of indexed Allocation).

- Unix follows I-node way of implementation in order to allocate disk-space to the files.
- Every file is associated with its own I-Node.
- Whenever the file is created then depending on the size of the file, file will be stored in the various DB addresses. like from Direct DBA to ..... so on.

Eg: $\boxed{FS \leq 10KB}$ Direct DBA

$\boxed{FS \leq 110KB}$ Single Indirect

[where DB size = 1KB and DBA has 100 Addresses]

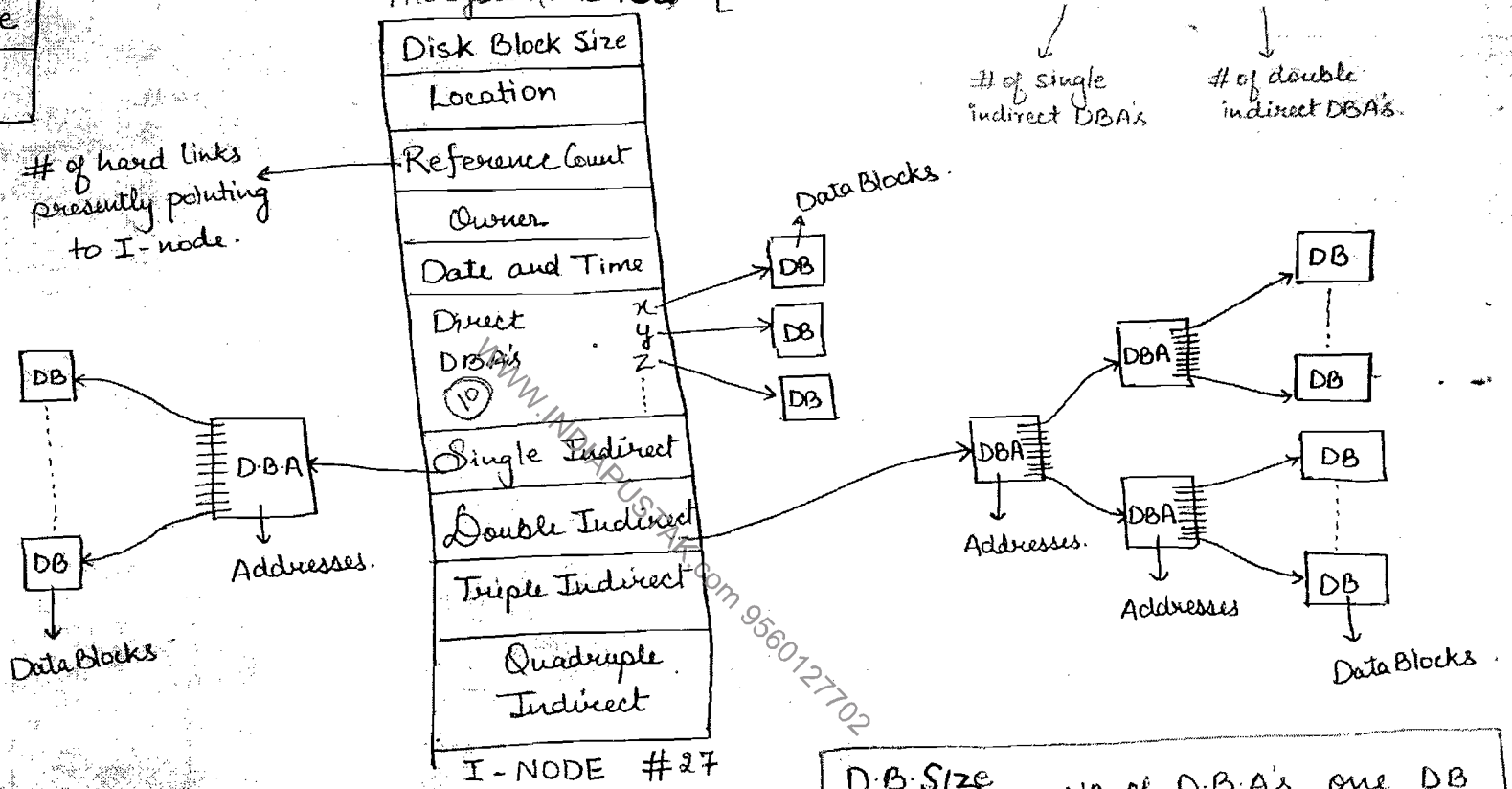
Scanned by CamScanner

file	I-node
abc.doc	27

Total Size of the File System (maxFileSize) $= \left[\#\text{ of Direct DBA's} + \left(\frac{DB\ Size}{DBA}\right) + \left(\frac{DB\ Size}{DBA}\right)^2 + \left(\frac{DB\ Size}{DBA}\right)^3 + \dots \right] DB\ Size$

# of single indirect DBA's

# of double indirect DBA's



$\frac{D.B.Size}{DBA}$ = No. of D.B.A's one DB can contain

WWW.INDIAPUSTAK.com 9560127702

Scanned by CamScanner

Ques. Consider the system which has Unix-I node with 10 direct DBA, 1 single indirect DBA and 1 double indirect DBA and 1 triple indirect DBA. Disk block ~~add~~ size = 1kB and disk block address requires 32 bits than what is the Max. file size possible with the triple indirect ?

a). 2 GB b). 4 GB c). 8 GB d). 16 GB.

$$\left(\frac{DB\ Size}{DBA}\right)^3 * DB\ Size = \left(\frac{2^{10}}{2^2}\right)^3 * 2^{10} B = 2^{34} B$$

$$= \underline{16\ GB}..$$

Ans. 6 (113).

Direct DBA = 5

$$\left[5 + 3 * \left(\frac{400}{10}\right) + 2 * \left(\frac{400}{10}\right)^2\right] * 400B$$

$$\Rightarrow [5 + 120 + 3200] * 400B$$

$$\Rightarrow \underline{1330000\ B}.$$ (c) ✓

Ans. 7 (113)

$$\left(\frac{DB\ size}{DBA}\right)^4 * DB\ size = 4TB$$

$$\left(\frac{1K}{x}\right)^4 * 1KB = 4TB$$

$$\frac{2^{50}}{x^4} = 2^{42}$$

$$x^4 = 2^8$$

$$x = 2^2 = 4B$$

$$x = \underline{32\ bits}.$$ (d) ✓

WWW.INDIAPUSTAK.com 9560127702

Scanned by CamScanner

Disk Free Space Management

Size of the disk = 20MB

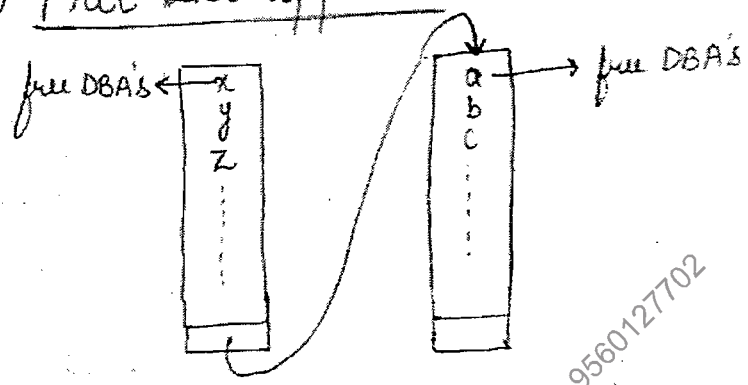
DB size = 1KB

DBA = 16 bits

# of disk blocks available on given disk –

$$\frac{\text{Size of disk}}{\text{DB size}} = \frac{20MB}{1KB} = 20K$$

① Free List approach :



- In this approach, some disk blocks are used just to store the free disk block addresses.

① # of DBA possible to store in one disk block :

$$= \frac{\text{DB size}}{\text{DBA}} = \frac{1KB}{2B} = 2^9 = 512$$

② 512 DBA's ⟷ we can store in 1 DB

to store 20K DBA ⟶ $\frac{20K}{512} \approx 1 = \underline{40}$

② Bit Map Approach :

10110010
0110 1101
01101011
10110101
00101101

Bit Map

Bit → 0 free, 1 busy

Free Disk Blocks : 1, 4, 5, 7

Busy : 0, 2, 3, 6

- In this approach every disk block will be mapped with 1 binary bit.

WWW.INDIAPUSTAK.COM 9560127702

Scanned by CamScanner

① # of DB we can map in 1 DB.

DB size = 1KB.

= 1K * 8 = 8K bits

② 8K DB we can map in ←——→ 1 DB

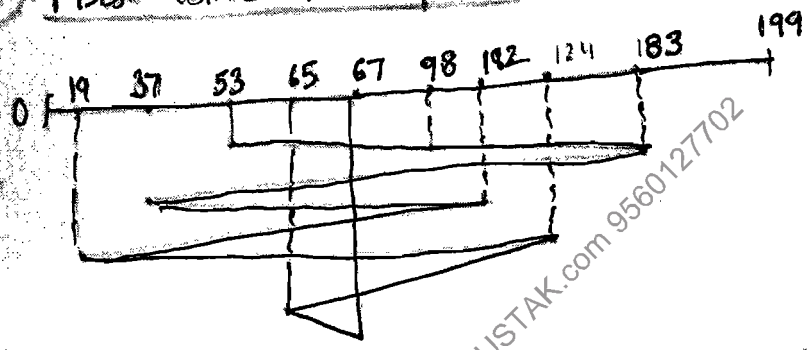
to map 20K DB's ——→ $\frac{20K}{8K} * 1 = 2.5 \approx \underline{3}$.

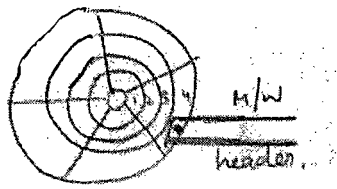
Disk Scheduling :

Goal: to minimise the avg seek time of disk.

eg: track requests → 98, 183, 37, 122, 14, 124, 65, 67.

① First Come First Serve





WWW.INDIAPUSTAK.com 9560127702

Total track movements made by H/W header (seek time):

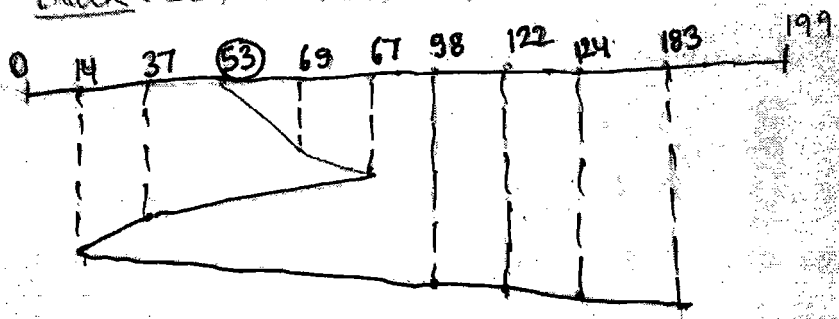
$(98-53) + (183-98) + (183-37) + (122-37) + (122-14) + (124-14) + (124-65) + (67-65)$

$= \underline{640}$

② Shortest Seek Time First (SSTF) (or)

Nearest track next.

track : 98, 183, 37, 122, 14, 124, 65, 67.



$= (67-53) + (67-14) + (183-14)$

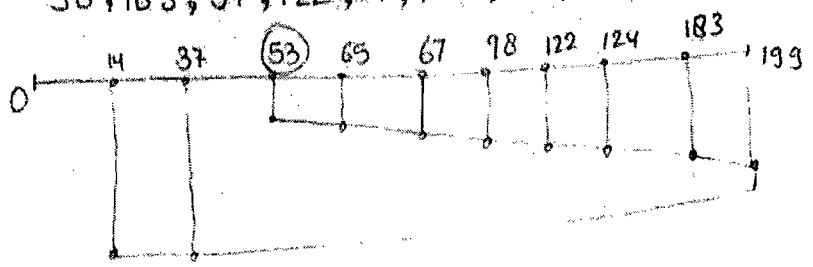
$= 14 + 53 + 169$

$= \underline{236}$

Scanned by CamScanner

③ Scan (Elevator):

98, 183, 37, 122, 14, 124, 65, 67

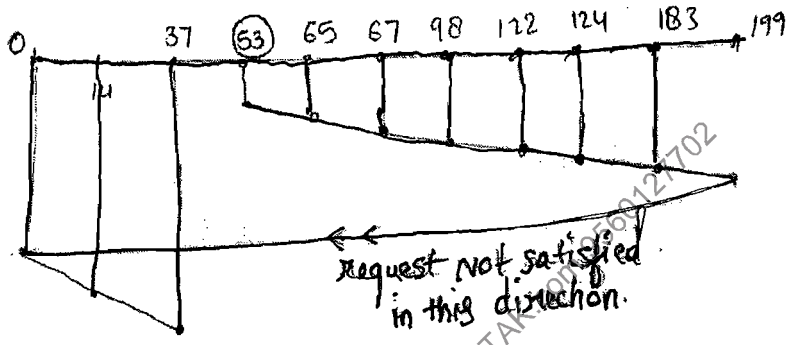


$= (199 - 53) + (199 - 14)$

$= 146 + 185$

$= \underline{331}$

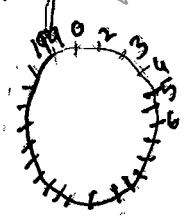
④ Circular Scan (C-scan):-



$(199 - 53) + (199 - 0) + (37 - 0)$

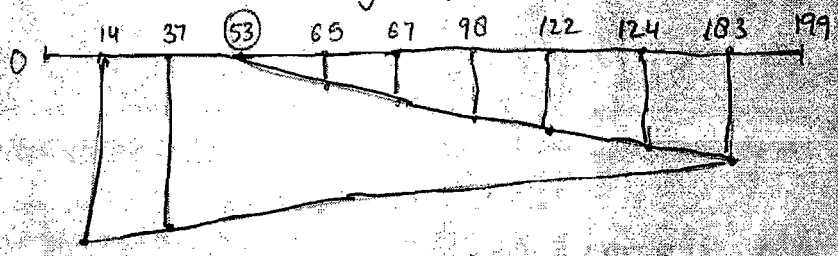
$= \underline{382}$

we are assuming that tracks are in circular manner (practically not possible)



⑤ Look

(Look for last pending request in the direction of r/w header)



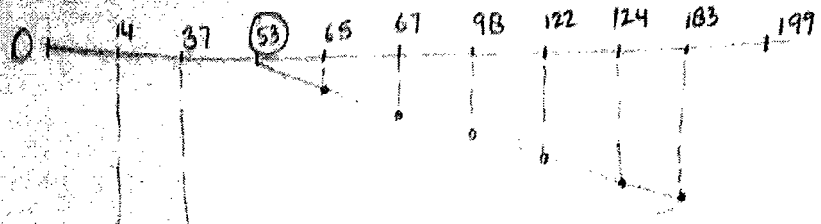
$(183 - 53) + (183 - 14) = \underline{299}$

difference b/w scan and look is that scan goes until last (199) & look doesn't

WWW.INDIAMUSTARDPRO.COM 9012702

Scanned by CamScanner

(6) C-Look



Request Not satisfied in this direct^n (including it bcz practically Not possible to jump directly 183 to 14).

$\Rightarrow (183-53)+(183-14)+(37-14)$

$\Rightarrow 130+169+23$

$\Rightarrow \underline{322}$.

WWW.INDIAPUSTAK.com 9560127702

Scanned by CamScanner

WWW.INDIAPUSTAK.com 9560127702