

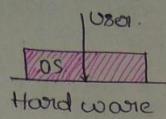
OPERATING SYSTEMS

1. PROCESS MANAGEMENT

ekexam.com

I. INTRODUCTION TO OS

→ Operating System is an interface between user and Hardware.



→ Operating System acts as Resource Allocator. (Resource means anything like CPU, memory, printers), OS takes care of Resource Allocation (takes the responsibility to allocate CPU to some process, I/O to some process etc). Some Resources cannot be shared.

→ OS acts as manager ⇒ keeps track of the resources that are allocated to particular process (Memory, process, files, security are managed by OS).

GOALS:

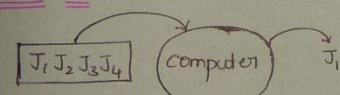
Primary goal = Convenience

Secondary goal = Efficiency.

Types of operating system

Batch OS, Multiprogramming, Multitasking,
multiprocess, Realtime.

Batch OS



⇒ processes need two types of times they are CPU time and I/O time.

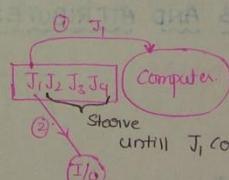
⇒ After completion of one job the other will start.

⇒ Not useful for interactive Applications.

⇒ Starvation.

⇒ Less Throughput

⇒ No preemption.

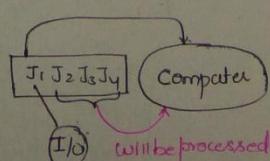


Multiprogramming

⇒ CPU will not be kept idle.

⇒ whenever a job needs I/O then the CPU takes the next process and process it.

∴ CPU will be busy all the time.



⇒ No starvation.

⇒ Efficiency is more.

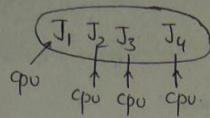
⇒ More throughput.

⇒ No preemption.

Multi Tasking:

⇒ The CPU will be multiplexing among the jobs without completing the job completely and that way interactivity can be improved.

⇒ Preemption is present.



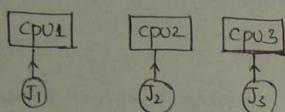
Multi processing:

⇒ Instead of single CPU we will have lots of CPUs in a single computer.

⇒ Many jobs can be processed simultaneously.

⇒ Parallelism is improved.

⇒ Throughput is more. (Throughput is the no. of jobs that could execute in unit time)



⇒ Dual core (2 CPUs) Quadcore (4 CPUs)

⇒ Each CPU can be used as we wish we could use CPU1 as Both processing, CPU2 can be used as multitasking etc.

Realtime:

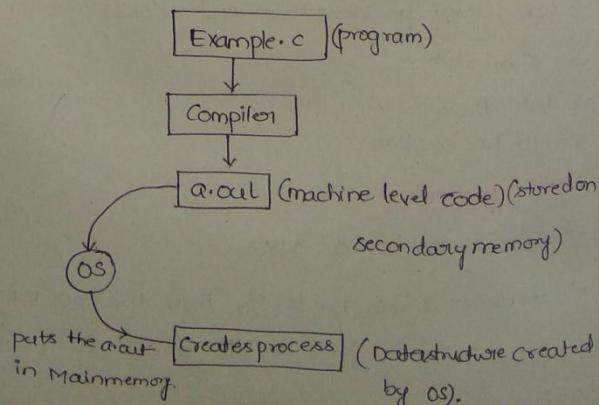
⇒ We will have jobs with deadlines.

⇒ Result produced after the deadline is waste

2. PROCESS, PCB AND ATTRIBUTES

Attributes of a process

- 1) process id
- 2) Program Counter
- 3) process state
- 4) priority
- 5) General purpose Registers
- 6) List of open files
- 7) List of open devices
- 8) Protection.



The operating system

- the main memory

⇒ The execution starts if cross

→ process I

→ program C

→ process R

→ priority

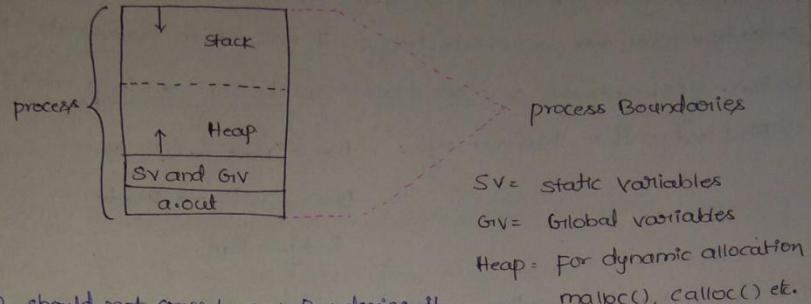
→ GPR :

→ list of open files

→ All this info

⇒ Every process

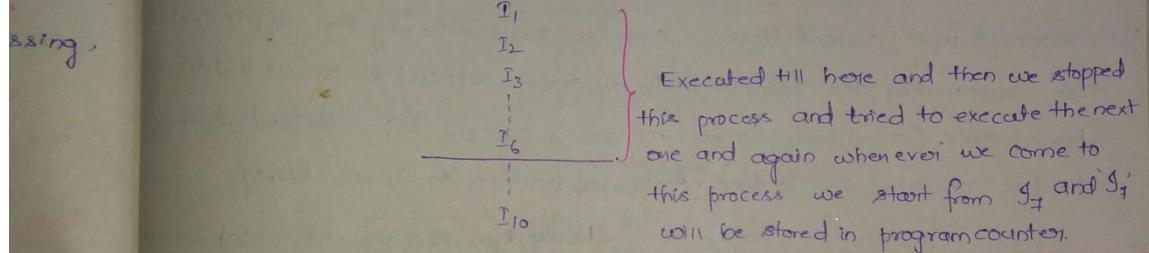
(2) The operating system takes the a.out file and creates a structure like this in the main memory.



⇒ The execution should not cross process boundaries, if it crosses we get "Segmentation fault".

→ process Id: Unique no. that is given to a process to identify it.

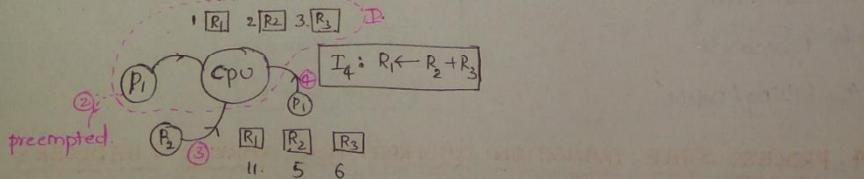
→ program counter: Stores the starting point of next instruction.



→ process state: New, Ready, Running, Blocked

→ priority: Number that is assigned to the process whenever it is created.

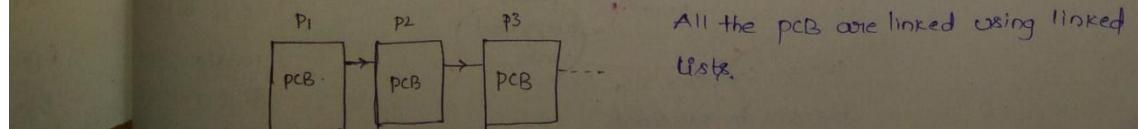
→ GPR:



→ list of open files: While executing some files are open for reading and some for writing we should remember which files we have read/ which files we have written.

→ All this information will be present in process control block

→ Every process gets its own PCB.



3. PROCESS STATES AND MULTI PROGRAMMING

(4)

States of process:

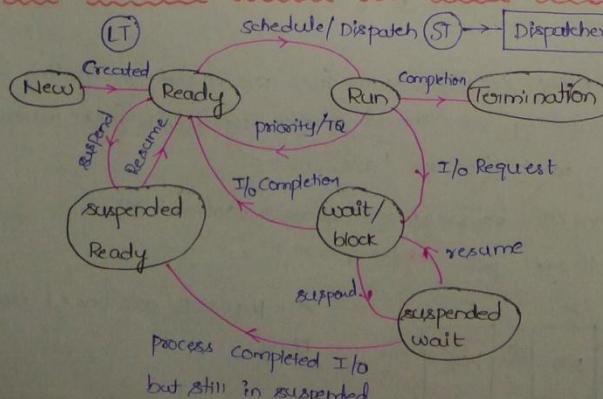
- 1> New = program present in Secondary Memory and is Ready to pick by the OS is called "New" state.
- 2> Ready = whenever we create process it will be in Ready state (process is in MainMemory).
- 3> Run - Mainmemory - Only one process will be running in a single CPU.
- 4> Block and wait - Mainmemory - Run → Block / wait state
Block / wait → Ready
Ready → Run
- 5> Termination / completion = Context will be deleted, Every trace of process will be deleted.
- 6> Suspend Ready - when the main memory is full and a new process having a highest priority has arrived then a process in the main memory should be suspended and room must be made for new process.
- 7> Suspended wait / suspend block = same as suspend ready the diff is instead of suspending the process in Ready state, suspend the process that are in block state. (Secondary Memory) - when a process is in suspend wait and it has completed its I/O then it makes transition to Suspend Ready.

⇒ Degree of State at r

⇒ Short Term deciding for moving "Context-

⇒ The medium is the memory

4. PROCESS STATE TRANSITION DIAGRAM AND VARIOUS SCHEDULERS



TQ = Time Quantum.

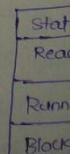
LT = Long Term Scheduler

ST = Short term scheduler

MT = Medium Term Scheduler

5. QUESTION

Consider a syst



Degree of multiprogramming = No. of processes that can be present in the Ready state at maximum. (This factor is decided by Long Term scheduler) (5)

"New" state Memory → Short Term scheduler decides which process must be executed next and after deciding it will call "Dispatcher". Now, Dispatcher is the glo that is responsible for moving the process from Ready- Run and Run-Ready state. It is called "Context- switching".

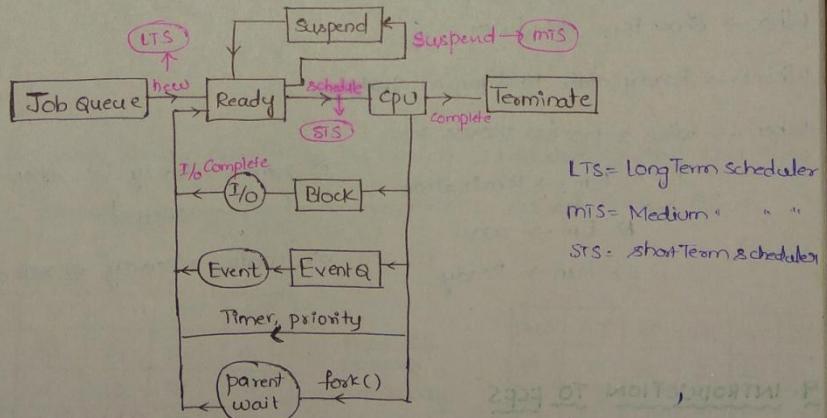
→ The medium/mid Term Dispatcher is responsible for swapping. Swapping is the process of moving the process from Main memory to secondary memory and from secondary memory to Main memory.

Long Term scheduler - Related to overall performance

Short Term scheduler - Related to context switching time

Medium Term Scheduler - Impacts Swapping Time

5. PROCESS QUEUES



6. QUESTION ON PROCESS STATES

Consider a system with 'N' CPU processors and 'M' processes then.

State	min	max
Ready	0	M
Running	0	N
Block	0	M

7. VARIOUS TIMES RELATED TO PROCESS

1) Arrival time : The time at which the process enters "Ready Queue".

(6)

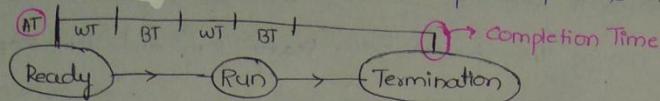
Gantt Chart

2) Burst time : The amount of CPU time required by a process to finish.

3) Completion time : The time at which the process finishes execution.

→ In case

lower p



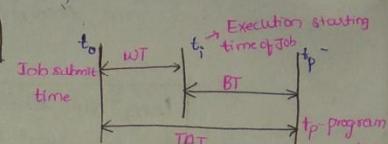
$$[CT - AT = BT + WT] \Rightarrow [CT - AT = TAT]$$

4) Turnaround Time : The difference between the completion and arrival times is called TAT.

$$TAT = CT - AT$$

5) Waiting Time :

$$WT = TAT - BT$$



6) Response time : What is the first time the process hits the CPU.

⇒ IN the waiting ti

8. CPU SCHEDULING

Picking the process from ready state and giving it to CPU is called CPU scheduling.

Who → Short-term scheduler

CONVOY

⇒ when a p
all the

Where → Ready state to Running state

When → when a process moves from

I. a) Run → Termination

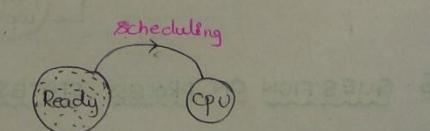
2) New → Ready i.e. when a process is just created

b) Run → wait

3) wait → Ready ⇒ Not all time → Based on Priorities.

c) Run → Ready

↓



9. INTRODUCTION TO FCFS

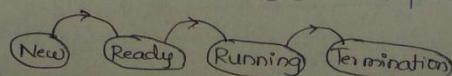
Criteria : Arrival Time

Gantt Chart

Mode : Non-preemptive

PNO	AT	BT
1	0	4
2	1	3
3	2	1
4	3	2
5	4	5

⇒ The mode is Non-preemptive because when we give a job to CPU we will not forcibly pull it from CPU. We wait until the job is finished. So the mode is Non-preemptive.



PNO
1
2
3

P1
20

$$\text{Avg WT} = \frac{19}{3}$$

$$\text{Avg TAT} = \frac{80}{3}$$

Gantt chart :

P ₁	P ₂	P ₃	P ₄	P ₅
0	4	7	8	10

(6)

(7)

h
ecution

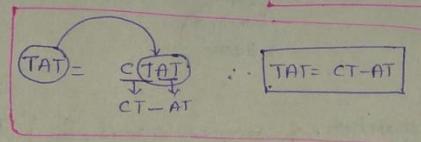
→ In case if 2 processes have same arrival times then pick the process with lower process id / PNO.

PNO	CT	TAT = CT - AT	WT = TAT - BT
1	4	4	0
2	7	6	3
3	8	6	5
4	10	7	5
5	15	11	6

$$\text{Avg. TAT} = \frac{4+6+6+7+11}{5}$$

$$\text{Avg. WT} = \frac{0+3+5+5+6}{5} = 4$$

⇒ IN the case of Non-preemptive version Both the Response time and waiting time of a process is same ⇒ Response time = waiting Time



10. CONVOY EFFECT

⇒ when a process with heavy Burst time is being scheduled by the CPU then all the other process are going to starve, this is called "convoy Effect."

PNO	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	2	22	21	19
3	1	1	23	22	21

Reschedule
In this way

PNO	AT	BT	WT	TAT	CT
P ₁	1	20	2	22	23
P ₂	0	2	0	2	2
P ₃	0	1	2	3	3

Gantt chart

P ₁	P ₂	P ₃
0	20	22

$$\text{Avg WT} = \frac{19+21}{3} = \frac{40}{3}$$

$$\begin{aligned} \text{Avg TAT} &= \frac{20+21+22}{3} = \frac{63}{3} \\ &= 21 \end{aligned}$$

Gantt chart

P ₂	P ₃	P ₁
0	2	3

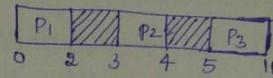
$$\text{Avg. WT} = \frac{0+2+2}{3} = \frac{4}{3}$$

∴ WT has drastically reduced

11. FCFS EXAMPLE

PNO	AT	BT	CT	TAT	WT
1	0	2	2	2	0
2	3	1	4	1	0
3	5	6	11	6	0

Gantt chart



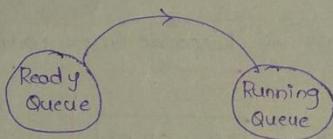
→ Here no process is waiting for CPU. In fact CPU is waiting for the processes.

⇒ Queue is the simplest Data structure that is used to implement FCFS.

12. FCFS WITH OVERHEAD

⇒ The context switching time "δ" will be given.

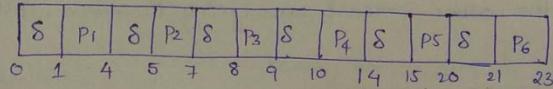
⇒ The context switching time (δ) is when a process is running, we are going to stop it and schedule the other process. In between many things will happen now,



{ scheduler → dispatcher → process is ready for execution } Time taken for this entire process is called context-switching time

PNO	AT	BT
1	0	3
2	1	2
3	2	1
4	3	4
5	4	5
6	5	2

Gantt chart



⇒ Before the first process is called the scheduler will be called which in turn calls the dispatcher and the process is made available for execution. So context-switching occurs before every new process is brought to execution.

so the total life cycle takes 23ms to complete out of which 6ms is wasted for context switches and hence the Alg's efficiency decreases.

$$\text{Inefficiency} = \frac{6 \times 1}{23} \times 100$$

$$\text{Efficiency } \eta = (1 - \frac{6}{23}) \times 100$$

⇒ SJF

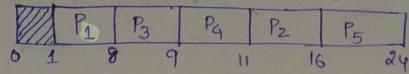
13. INTRODUCTION TO SJF

Criteria - Burst time

Mode = Non-preemptive

PNO	AT	BT	CT	TAT	WT
1	1	7	8	7	0
2	2	5	16	14	9
3	3	1	9	6	5
4	4	2	11	7	5
5	5	8	24	19	11

Gant chart



⇒ Shortest Job first Algorithm schedules the job having the least Burst time among the "AVAILABLE PROCESS."

⇒ victim of convoy effect.

⇒ The datastructure used is "Minheap." (Root will be the process with least bursttime)

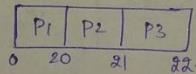
14. ANALYSIS OF SJF

⇒ SJF cannot be implemented practically.

PNO	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	1	21	20	19
3	2	1	23	21	20

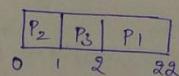
$$\text{Avg. WT} = \frac{29}{3} = 13 \text{ ms.}$$

$$\text{Throughput} = \frac{3}{22}$$



PNO	AT	BT	CT	TAT	WT
1	2	20	22	20	0
2	0	1	1	1	0
3	1	1	2	1	0

Gant chart:



$$\text{Avg. WT} = 0.$$

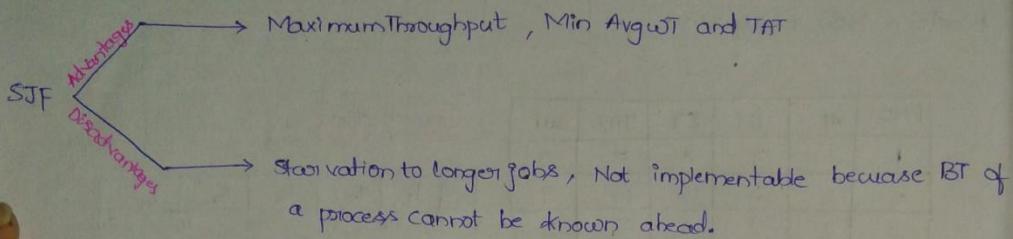
$$\text{Throughput} = \frac{3}{22}$$

→ No of process
→ Total schedule time

⇒ SJF gives the best throughput than any other scheduling Algorithms.

15. SJF WITH PREDICTION OF BT

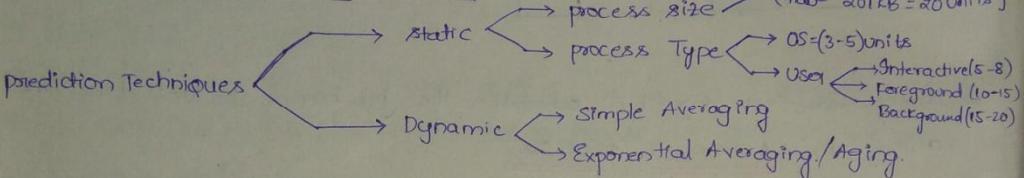
(10)



Solution:

SJF with predicted BT's.

Prediction Techniques



One of the Best method available for Burst Time predicting is "Simple Averaging"

Simple Averaging

→ Given 'n' processes (P_1, P_2, \dots, P_n)

→ Let T_i be the actual BT.

→ Let T'_i denote the predicted BT

$$T'_{n+1} = \frac{1}{n} \sum_{i=1}^n t_i$$

Exponential Average / Aging

$$T'_{n+1} = \alpha t_n + (1-\alpha)T'_n \quad (0 \leq \alpha \leq 1) \quad \left\{ \begin{array}{l} t_n = \text{Actual BT} \\ T'_n = \text{predicted BT} \end{array} \right. \quad \alpha = \text{smoothing factor}$$

Ex: $\alpha = 0.5$, $T'_1 = 10$ actual BT $(t_1, t_2, t_3, t_4) = (4, 8, 6, 7)$ then $T'_5 = ?$

$$\text{Now, } T'_5 = (0.5)t_4 + (1-0.5)T'_4$$

$$T'_5 = (0.5)7 + (0.5)(6.75)$$

$$T'_5 = 6.875$$

$$\text{Now, } T'_4 = (0.5)T'_3 + (0.5)T'_3$$

$$T'_3 = (0.5)t_2 + (0.5)T'_2$$

$$T'_2 = (0.5)t_1 + (0.5)T'_1$$

$$T'_1 = 10$$

$$\text{Now, } T'_2 = (0.5)(4) + (0.5)10 \\ = 2 + 5 = 7$$

$$T'_3 = (0.5)8 + (0.5)7 \\ = 4 + 3.5 = 7.5$$

$$T'_4 = (0.5)6 + (0.5)7.5 = 6.75$$

16. INTRO

SRTF =

Criteria =

MODE : P

PNO
1
2
3
4
5
6

Gantt chart

⇒ Cannot

⇒ Any sd
because

17. Example

PNO
1
2
3

18. Example

PNO
1
2
3
4

16. INTRODUCTION TO SRTF

SRTF = Shortest Remaining Time first.

Criterias = BT's.

Mode : pre-emptive

BT of

= 20 units }

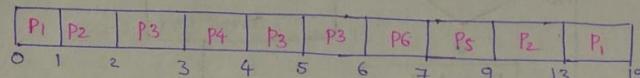
= 20 units }

arrivals - 8
ground (10-15)
ground (15-20)

PNO	AT	BT	CT	TAT	WT
1	0	7	19	19	12
2	1	5	13	12	7
3	2	3	6	4	1
4	3	1	4	1	0
5	4	2	9	5	3
6	5	1	7	2	1

PNO	BT
1	7 6 0
2	5 4 0
3	8 2 0
4	1 0
5	7 0
6	1 0

Gantt chart :



aging"

⇒ Cannot be practically implemented.

⇒ Any scheduling algorithm related to BT cannot be implemented practically because BT cannot be predicted accurately.

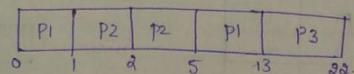
17. EXAMPLE ON SRTF (SPP 2011)

factor }

PNO	AT	BT	CT	TAT	WT
1	0	9	13	13	4
2	1	4	5	4	0
3	2	9	22	20	11

What is the Avg. WT using SRTF?

$$\text{Avg. WT} = 15/3 = 5 \text{ ms.}$$



18. EXAMPLE ON SRTF (IITAE 2007)

T₃

2

T₁

s) 10

+

5) 7

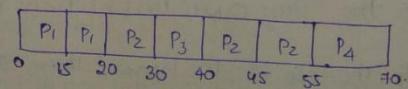
= 7.5

7.5 = 6.75

PNO	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	15	25	55	40	15
3	30	10	40	10	0
4	45	15	70	25	10

what is the WT of P2?

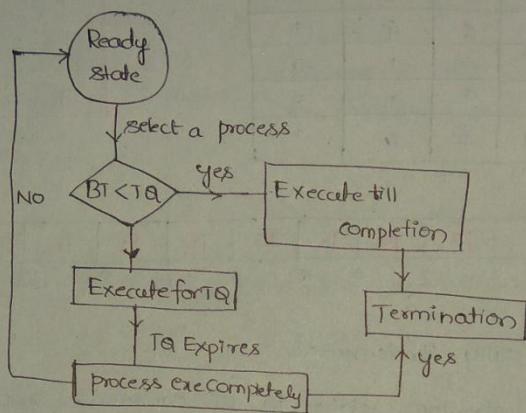
Gantt chart



$$\therefore \text{WT of P}_2 = 15.$$

19. ROUND ROBIN ALGORITHM

- ⇒ Many of the operating systems in practice use Round Robin scheduling.
- ⇒ Not depending on BT
- ⇒ Simple Data structure: Queue
- ⇒ Each process is executed for TQ amount of time ($TQ = \text{Time Quantum}$)
- ⇒ Less starvation problem.



20. ROUND ROBIN EXAMPLE 1

CRITERIA: $TQ + AT$ (First come first serve)

ROUND ROBIN: PREEMPTIVE

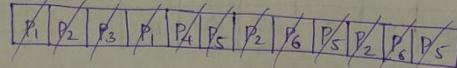
$TQ = \text{Max amount of time for}$

which a process is allowed to execute when it is scheduled.

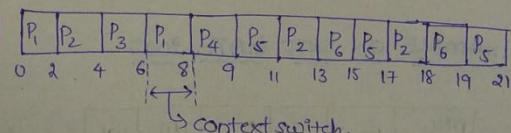
PNO	AT	BT
1	0	4
2	1	5
3	2	2
4	3	1
5	4	6
6	6	3.

$TQ = 2$

Queue:



Gant chart:



⇒ As the Time Quantum [↑] then no. of context switches [↓]

⇒ If the TQ is too large then starvation occurs.

21. ROUND ROBIN Example 2

TQ=3.

PNO	AT	BT	CT	TAT	WT	RT
1	5	5	32	27	22	10
2	4	6	27	23	17	5
3	3	7	33	30	23	3
4	1	9	30	29	20	0
5	2	2	6	4	2	2
6	6	3	21	15	12	12

process Queue: $\boxed{P_4 | P_5 | P_3 | P_2 | P_4 | P_1 | P_6 | P_5 | P_2 | P_4 | P_1 | P_3}$

Gantt chart: $\boxed{\text{---} | P_4 | P_5 | P_3 | P_2 | P_4 | P_1 | P_6 | P_5 | P_3 | P_2 | P_4 | P_1 | P_3}$

PNO	BT
1	5 \neq 0
2	6 \neq 0
3	7 \neq 0
4	9 \neq 0
5	2 \neq 0
6	3 \neq 0

Response Time (RT): The diff b/w the time at which a job is submitted to CPU and the time which CPU starts scheduling it.

Ex: P_1 has been given to CPU at '5' (AT) but CPU started ' P_1 ' at 15 (Look at Gantt chart)
 $\therefore (RT)_{P_1} = 15 - 5 = 10$.

TQ CS RT.
 \uparrow \downarrow \uparrow .
 \downarrow \uparrow \downarrow .

If $TQ = \infty$ then, Round Robin will become FCFS.

22. ROUND ROBIN EXAMPLE-3

Consider 4 Jobs P_1, P_2, P_3 and P_4 arriving in Ready Queue in the same order at time $t=0$. If BT requirements of these jobs are 4, 1, 8, 1 respectively, what is the completion time of P_1 , assuming Round Robin with $TQ=1$?

PNO	AT	BT	CT
P_1	0	4	9
P_2	0	1	2
P_3	0	8	14
P_4	0	1	4

Queue: $\boxed{P_1 | P_2 | P_3 | P_4 | P_1 | P_3 | P_1 | P_2 | P_3}$

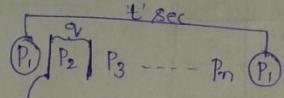
Gantt chart: $\boxed{P_1 | P_2 | P_3 | P_4 | P_1 | P_3 | P_1 | P_3 | P_1 | P_3}$

$\therefore (CT)_{P_1} = 9$.

Q3. ROUND ROBIN EXAMPLE 4

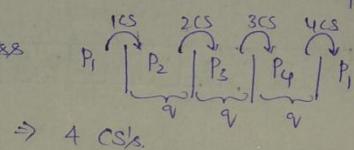
Consider 'n' processes sharing the CPU in RR fashion. If the context switching time is 's' units. what must be the time quantum 'Q' such that the no. of context switches are reduced, but at the same time each process is guaranteed to get the turn at the CPU for every 't' seconds?

let the processes be



Context-Switch time = Taken time to shift from one process to another process.

Now, let's analyze for 4 processes



$\Rightarrow 4 \text{ CSs}$

$\Rightarrow 3 \text{ quanta}$.

∴ generalizing the above ex: $n(s) + (n-1)s \leq t$

$$\Rightarrow q \leq \frac{t - ns}{(n-1)}$$

Q4. LONGEST JOB FIRST

→ process having longest BT gets scheduled first

CRITERIA: BT

MODE: Non-preemptive

PNO	AT	BT	CT	TAT	WT	RT
1	0	3	3	3	0	0
2	1	2	20	19	17	19
3	2	4	18	16	12	12
4	3	5	8	5	0	0
5	4	6	14	10	4	4

PNO
1
2
3
4

The

Q5. LONGEST REMAINING TIME FIRST

PNO
1
2
3

The Gantt chart is: $\begin{array}{|c|c|c|c|c|c|} \hline & P_1 & P_4 & P_5 & P_3 & P_2 \\ \hline 0 & 3 & 8 & 14 & 18 & 20 \\ \hline \end{array}$

PNO
1
2
3

Q5. ROUND ROBIN Example
CRITERIA

- HRRN
- longer
- Mode:

PNO	AT	BT	CT	TAT	WT	RT
1	1	2	18	17	15	0
2	2	4	19	17	13	0
3	3	6	20	17	11	0
4	4	8	21	17	9	0

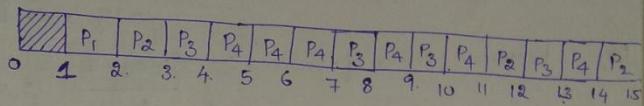
(14)
itching time
context
to get the

PNO	AT	BT
1	1	2 $\times 10$
2	2	4 $\times 3 \times 10$
3	3	6 $\times 5 \times 3 \times 10$
4	4	8 $\times 7 \times 5 \times 3 \times 2 \times 10$

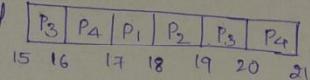
(15)

one process

The Gantt chart will be



Gantt chart continued



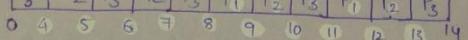
26. LONGEST REMAINING TIME FIRST GATE 2006 QUESTION

PNO	AT	BT	CT	TAT
1	0	2	12	12
2	0	4	13	13
3	0	8	14	14

what is the Average TAT using LRTF?

PNO	AT	BT
1	0	2 $\times 10$
2	0	4 $\times 3 \times 10$
3	0	8 $\times 3 \times 10$

Gantt chart:



$$\text{Avg. TAT} = \frac{12+13+14}{3} = \frac{39}{3} = 13 \text{ ms.}$$

27. HIGHEST RESPONSE RATIO NEXT

CRITERIA: Response Ratio (RR) = $\frac{W+S}{S}$

W = waiting time for a process so far

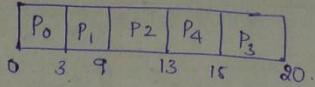
S = service time of a process or BT.

→ HRRN not only favours shorter jobs but also limits the waiting time of longer jobs.

→ Mode: Non-preemptive.

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

Gant chart:



30. PRI

(b)

Now, at time $t=9\text{ms}$.

$$RR_2 = \text{Response ratio of process 2} = \frac{(9-4)+4}{4} = 2.25$$

$\rightarrow P_2$ arrived at 4 and now the time is 9 so the waiting time till now is $(9-4) = 5$

$$RR_3 = \frac{3+5}{5} = 1.6$$

$$RR_4 = \frac{1+2}{2} = 1.5$$

more RR value so P_2 will be executed after 9ms of time.

Gant c

\Rightarrow Run

At time $t=13\text{ms}$

$$RR_3 = \frac{7+5}{5} = 2.4$$

$$RR_4 = \frac{5+2}{2} = 7/2 = 3.5 \rightarrow P_4 \text{ will be executed after } t=13\text{ms.}$$

28. PRIORITY SCHEDULING

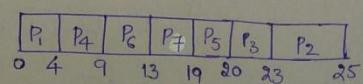
Priority

- static (Doesnot change throughout the execution of process)
- pre-emptive
- Non-preemptive
- Dynamic (changes at Regular intervals of time)

29. NON-PREEMPTIVE PRIORITY SCHEDULING

PNO	Priority	AT	BT	CT	TAT	WT	RT
1	2 (1)	0	4	4	4	0	0
2	4	1	2	25	24	22	22
3	6	2	3	23	21	18	18
4	10	3	5	9	6	1	1
5	8	4	1	20	16	15	15
6	12 (11)	5	4	13	8	4	4
7	9	6	6	19	13	7	7

Gant chart:



34. MU

Read

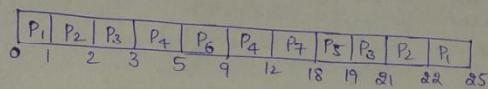
30. PRE-EMPTIVE PRIORITY SCHEDULING

PNO	Priority	AT	BT	CT	TAT	WT	RT
1	2	0	4	25	25	21	0
2	4	1	2	22	21	19	0
3	6	2	3	21	19	16	0
4	10	3	5	12	9	4	0
5	8	4	1	19	15	14	14
6	12	5	4	9	4	0	0
7	9	6	6	18	12	6	6

Max = 5
 $(9-4)!!.$

executed

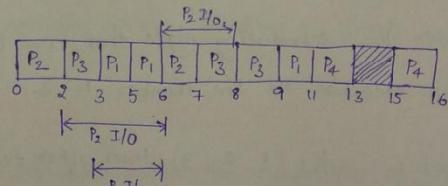
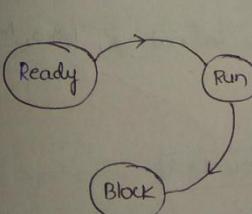
Gant chart:



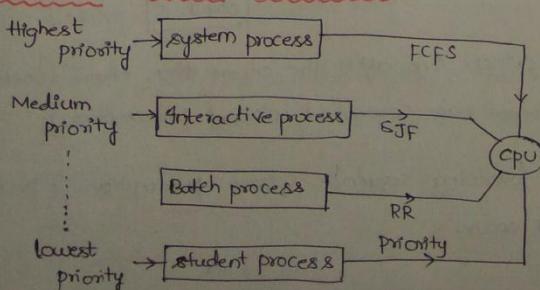
⇒ Run the process until a process with higher priority appears.

31. SRTF WITH PROCESS CONTAINS CPU AND I/O TIME EXAMPLE 1

PNO	AT	BT	IOBT	BT	IOBT = I/O Burst Time
1	0	(3	2	2)	= 5 (sum of CPU BT's).
2	0	(2	4	1)	= 3 (" " " ")
3	2	(1	3	2)	= 3 (" " " ")
4	5	(2	2	1)	= 3 (" " " ")



34. MULTILEVEL QUEUE SCHEDULING



⇒ There will be starvation

for lower level queues.

For various types of process

Various scheduling algos can be applied, this is the advantage.

④
Min no of

2. PROCESS SYNCHRONISATION

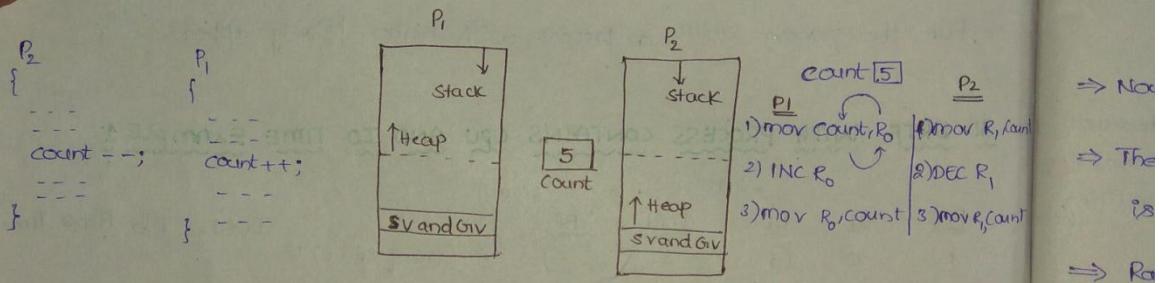
(8)

I. NEED FOR SYNCHRONISATION

Interprocess communication and synchronisation

⇒ whenever two processes need to communicate they need to communicate using shared memory, whenever they communicate using shared memory there may be some problem / some inconsistency.

⇒ consider two processes P_1 and P_2 in the same system need to access the shared variable "count" and P_1 increments the count value and P_2 decrements the count value.



The 1st instruction in the Assembly language code is "Reading" ($\text{mov}, \text{count}, R_0$)
2nd instruction is " " " " " " " " updating" (INC, R_0)
3rd instruction is " " " " " " " changing" ($\text{mov} R_0, \text{count}$)

Now consider the scheduling algos that are preemptive in nature, Now, let us say P_1 executes its line numbers 1, 2, 3.

P_1 : (1 2 3) then after execution $\boxed{\text{Count} = 6}$

P_2 : (1 2 3) then Initially $\text{Count} = 6$ — 1st statement

$\text{Count} -- = 5$ — 2nd statement
 $\boxed{\text{Count} = 5}$ — printed

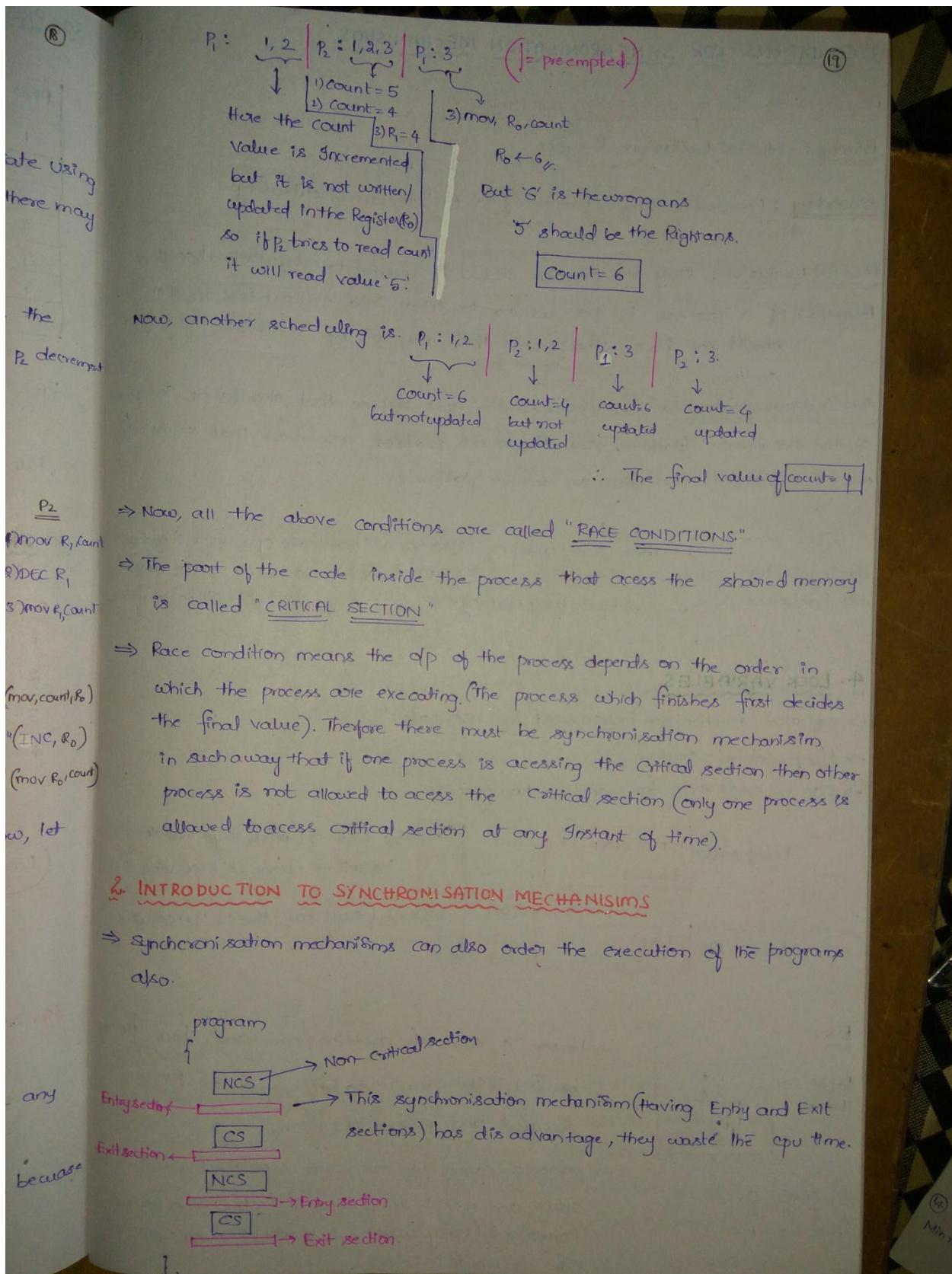
If these two processes occur one after the other then there won't be any inconsistency (If the processes are executed without preemption)

⇒ Now, consider the following sequence where preemption is present because of which some fault occurs.

2. INT
⇒ Sync
also

Entry section

Exit section



3. CONDITIONS FOR SYNCHRONISATION MECHANISMS

(2)

Now,

Requirements for synchronization mechanisms

Primary: Mutual Exclusion, Progress

Secondary: Bounded waiting, portability (or) Architectural Neutrality.

1. while
Entry

Exit 4. to

Mutual Exclusion: Only one process should be allowed to enter Critical section

Progress: If a process is not willing to enter into critical section then it should not block other process that are willing to access the critical section.

Architectural Neutrality: whatever solution you propose that should be independent of the hardware /platform. You should not propose a solution that runs only on one platform and fails on another platforms.

Synchronization Mechanisms

with Busy waiting (A process will not leave CPU until scheduler pulls it out as a result all the remaining processes will be waiting).

without Busy waiting

⇒ No

1)

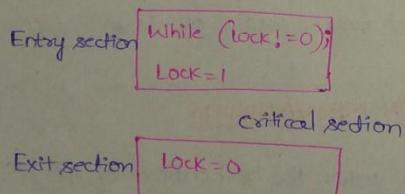
4)

4. LOCK VARIABLES

→ software mechanism implemented in user mode

→ Busy waiting solution

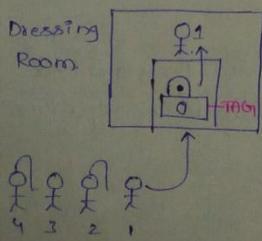
→ Can be used even for more than two processes



Initially Lock = 0 (which says that the CS is Vacant i.e. no one is executing CS)
Lock = 1 (says that the CS is occupied)

Now,

Ex:



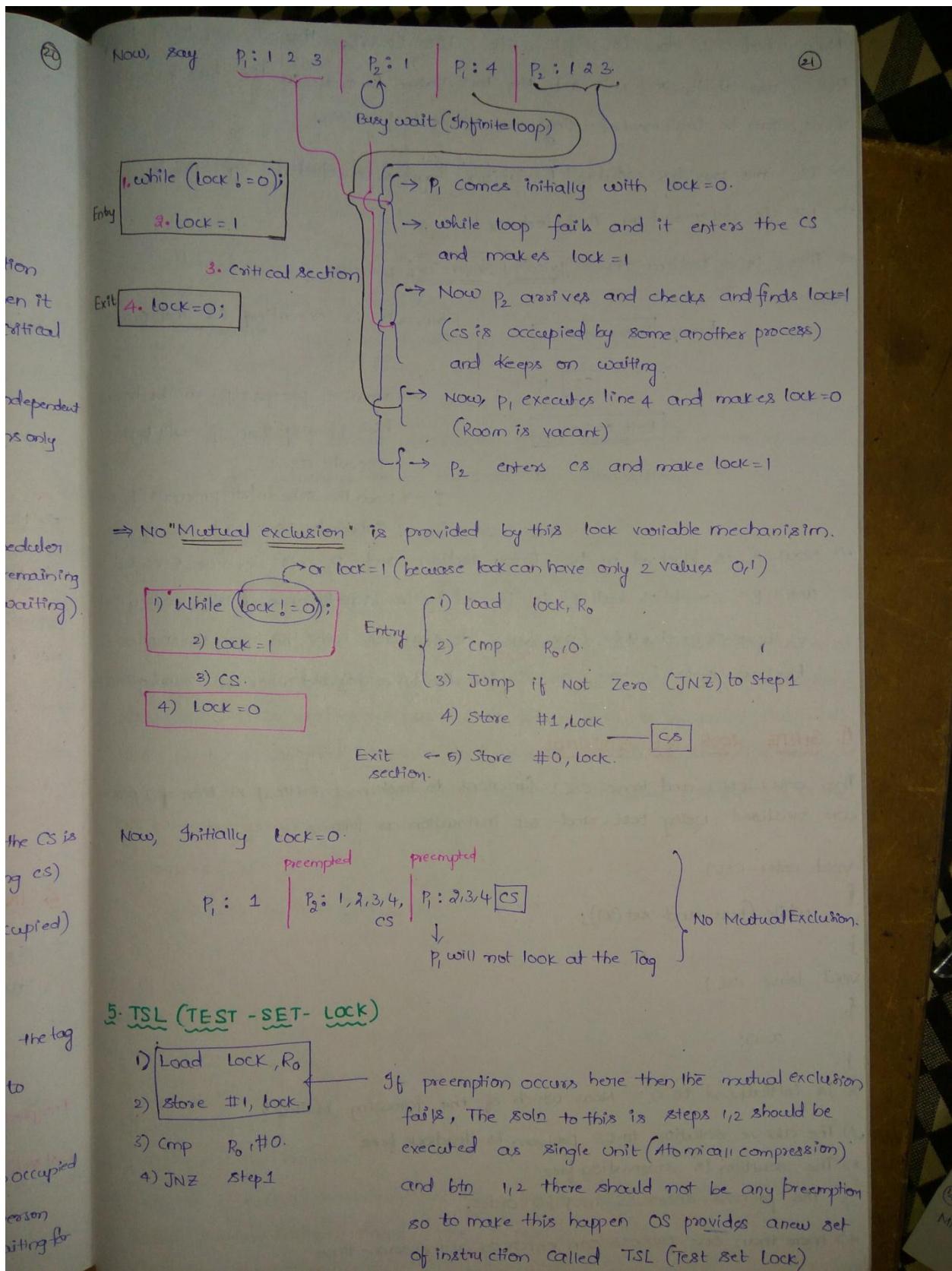
⇒ person 1 comes to dressing room and check the tag

he finds it is vacant so he changes the tag to occupied and enters into room.

⇒ person 2 comes to room and sees the tag is occupied state and keeps on waiting until the other person comes out (Busy waiting i.e. P₂ will be busy waiting for other person to come out).

5. TSL

- 1) Lo
- 2) St
- 3) Cr
- 4) Jn



Now, what TSL does is, if I write TSL lock, R_0 then it will load the lock value at R_0 and will set the lock value to one so the two instructions above can be implemented using one single instruction.

⇒ TSL provides Mutual Exclusion, Progress, Bounded waiting depends.

⇒ TSL is dependent on Architecture,

⇒ There is a problem called "Priority Inversion" problem

Entry [Entry section]

[CS] P₁

[Exit section]

Now, $\rightarrow P_1$ is executing CS and priority of $P_1 = 0$

\rightarrow Now a process $'P_2'$ will be having high Priority than $'P_1'$ will try to execute CS.

\rightarrow Now the Scheduler preempts $'P_1'$ and schedule P_2 .

\rightarrow Now, P_2 is blocked by the Entry section and P_2 will never be executed until $'P_1'$ completes but $'P_1'$ is blocked by the Scheduler and is in waiting state.

\therefore There is a deadlock (Spinlock) In deadlock both the process are in blocked state but one is in Ready state(P_2) and other is in running state(P_1)

GATE 2008 TSL QUESTION

The enter-CS() and leave-CS() functions to implement critical section of a process are realised using test-and-set instruction as follows:

```
Void enter-CS()
{
    while (test-and-set(x));
}

Void leave-CS()
{
    x=0;
}
```

x is initialised to '0'. Now, which of the following is True?

a) The above solution to CS problem is deadlock free

b) The solution is starvation free

c) The process enter CS in FIFO order.

d) More than one processes can enter CS at the same time

1) TSL
2) Starvation

{
1
2
3
4}

3) FIFO

4) If F1

4) false,

GATE 2

Fetch - And

the value

Value of

lock. 'L'

Corresponds

lock being

Acquire lock

while

}

Release lock

L=0

}

This impleme

a) fails as

b) fails as

c) wastes com

d) makes com

road the
no instruction

2) TSL never causes Deadlock (TRUE)

3) Starvation is possible

depends.

and priority

will be having
will try to

attempts 'P₁' and

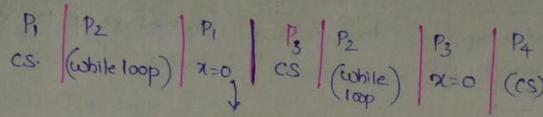
executed

waiting state

are in

running state (P₂)

on of a process



Here P₁ left
the CS now
instead of P₂, P₃ has entered
the CS (because it has the liberty
to execute (since x=0) and 'P₂' will
be waiting.)

P₂ will be starving all the time

3) FIFO Not possible (same explanation given above P₂ is never executed).

(If FIFO is guaranteed there won't be any starvation)

4) False, TSL guarantees Mutual exclusion.

7. GATE 2012 TSL QUESTION

Fetch-And-Add (x, i) is an atomic Read-modify-write instruction that reads the value of memory location X, increments the value by 'i' and returns the old value of 'X'. It is used in the pseudo code shown below to implement Busy wait lock. 'L' is an unsigned integer shared variable initialised to '0'. The value '0' corresponds to lock being available, while any non-zero value corresponds to the lock being not available.

Acquire Lock (L) {

 while (Fetch-And-Add (L, 1))

 L = 1;

}

Release Lock (L) {

 L = 0;

}

This implementation

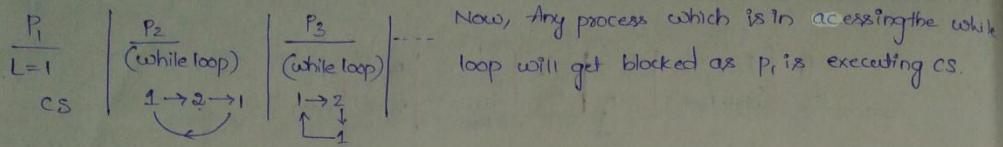
a) fails as 'L' can overflow

b) fails as 'L' can take non-zero value when the lock is actually available

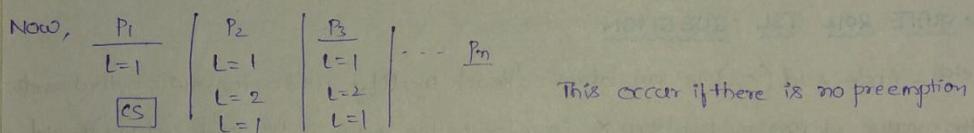
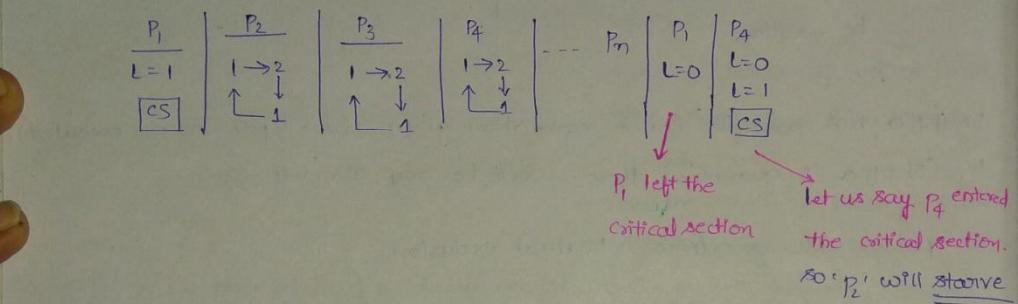
c) works correctly but may starve some process

d) works correctly without starvation.

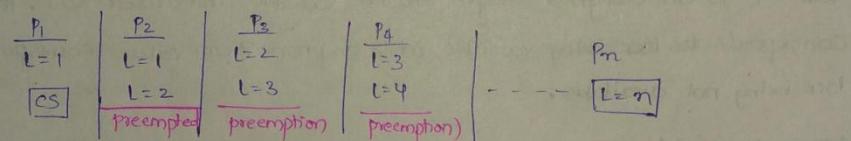
Now, Assume Initially $L=0$. Now, P_1 got the chance to execute CS. (24)



Now at some point of time ' P_1 ' leaves the critical section then, any of $P_2, P_3, P_4, \dots, P_n$ have the chance to enter the critical section, it need not be P_2 (FIFO order).



Now, if there is preemption at this point



$\therefore 'L'$ can overflow beyond its bounds.

\therefore This implementation fails ' L ' can overflow.

Now,

Aquire lock(L) {

1. while(fetch-and-add($L, 1$)) {

2. $L=1;$

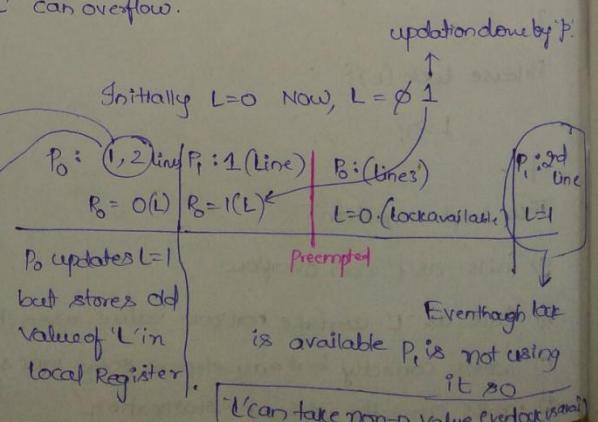
}

critical section

Release lock(L) {

3. $L=0;$

}



9. GATE

Consider as given randomly

Which

a) Mutual

b) Progr

S_1

T

F

F

Now, a

the criti

at at -

turn.

to ente

10. DISA

One of
Interrupt

Disab

CS

Enable

(24)

the while
ing CS.

(25)

g. GATE 2010 TSL QUESTION

Consider the methods used by process P_1 and P_2 to access \uparrow CS whenever needed as given below. The initial values of shared boolean variables S_1 and S_2 are randomly assigned

 P_2, P_3, P_4

Order:

Method used by P_1	Method used by P_2	
while ($S_1 == S_2$); CS $S_1 = S_2$;	while ($S_1 != S_2$); CS $S_1 = S_2$;	<ul style="list-style-type: none"> ⇒ If ($S_1 == S_2$) 'P_1' will fall in infinite loop ($; is imp$) ⇒ If ($S_1 != S_2$) 'P_2' will fall in infinite loop.

Which of the following is true?

- a) Mutual exclusion but not progress
- b) Progress but not ME
- c) Neither ME nor progress
- d) Both ME and progress.

P_4 entered
section.

Starve

option

S_1	S_2	
T	T	P_2 will be allowed to CS
T	F	P_1
F	T	P_1
F	F	P_2

At any point of time $S_1 \& S_2$ can have these combinations and at any instant only one process is accessing the CS
 \therefore ME is guaranteed

Now, assume both S_1 and S_2 are 'True' then ' P_2 ' has the chance to access the critical section, then assume ' P_2 ' is not interested to enter CS right now at this point of time ' P_1 ' tries to enter CS but ' P_1 ' is blocked because ' P_2 ' is turn. \therefore **progress is not guaranteed.** (A process (P_2) which is not interested to enter CS is not allowing another process to access it.)

which is interested to enter CS (P_1)

done by P_1

10. DISABLING INTERRUPTS

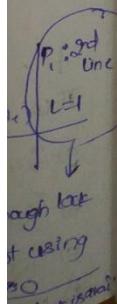
One of the way of handling the Synchronisation mechanism is "Disabling the Interrupts": A process should disable the interrupt before accessing the CS.

\Rightarrow If we disable interrupt preemption is not possible

\Rightarrow Both ME and progress is Guaranteed.

\Rightarrow Bounded waiting is not guaranteed unless you maintain the Queue.

\Rightarrow Architecture Dependent



Disabling the Interrupts.

CS

Enable the interrupt

II. TURN VARIABLE OR STRICT ALTERNATION METHOD

Strict Alteration Approach or Turn variable

→ S/o mechanism implemented at user mode

→ Busy waiting solution.

→ 2-process solution (P_0, P_1) or (P_i, P_j)

→ Mutual Exclusion is guaranteed

but progress is not guaranteed

→ Bounded wait is present (Guaranteed)

Busy waiting is absent

For process P_0 : (int turn)

Non critical section

while ($turn \neq 0$);

CS

$turn = 1;$

Non CS.

For process P_1 :

Non-CS.

while ($turn \neq 1$);

CS

$turn = 0;$

Non-CS.



① (Waiting for 1 Key).
② (Waiting for 2 Key).

After 1, 2 should go..

if 1 is not interested to go
and he is not giving key to 2,
then P_1 is blocking P_2 .

⇒ **[No progress.]**

⇒ Cannot be implemented for more than 2 processes.

Global Variable

P_0 (int turn) P_1

Non CS

while ($turn \neq 0$);

CS

$turn = 1;$

Non CS

Non-CS

while ($turn \neq 1$);

CS

$turn = 0;$

Non CS.

If $turn = 0 \Rightarrow P_0$ enters the CS.

If $turn = 1 \Rightarrow P_1$ enters the CS.

Now, In case of Lock Variables

LV

$L = 0 - P_0, P_1$ has chance
to enter CS.

$L = 1 - X$ No chance

Now, let's try by adding another variable

Now, let us consider another variable Interested[0] = T → I'm interested to enter CS

Interested[0] = F → I'm (P_1) not interested to enter CS.

Turn variable

0 - P_0 can enter the CS

1 - P_1 can enter the CS.

13. PETE

→ S/o

→ Busy

→ 2-p

→ St

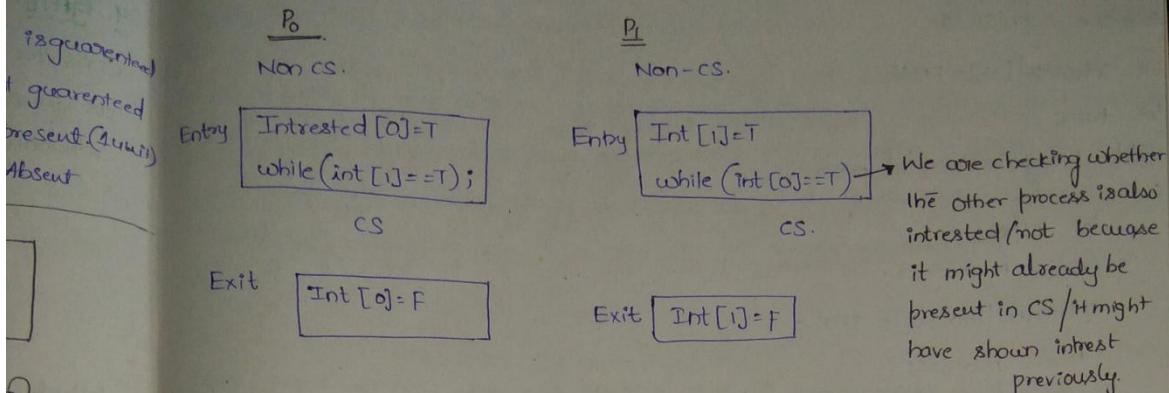
→ platform

→ No c

Entry

Critical

(26) Before entering the critical section a process should say whether it is interested/not interested to enter the critical section.



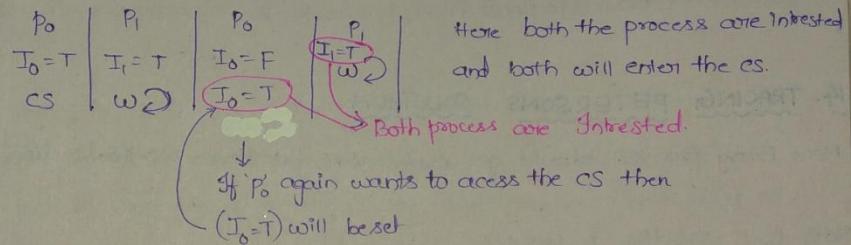
⇒ By the above manipulations ME, progress is guaranteed; Bounded wait (^{Not} possible).

Deadlock is possible

should go to
ring Key to '2'
Ring 'P₂'.

CS.

CS.



12. INTERESTED VARIABLE

Same concept in video 11 (Repeated).

13. PETERSON SOLUTION

- que mechanism at user mode
- Busy waiting solution
- 2-process solution (P_0, P_1)
- It uses (turn+interested) variable
- platform independent
- No overhead to the operating system (Everything executes in user mode)

Entry

Critical section

Exit

```

#define N 2
#define TRUE 1
#define FALSE 0

int interested[N] = FALSE
int turn;
void Entry-Section(int process)
{
    int other;
    other = !process;
    interested[process] = TRUE;
    turn = process;
    while (interested[other] == TRUE && TURN == process);
}

void Exit-Section(int process)
{
    interested[process] = FALSE;
}

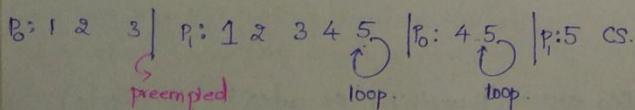
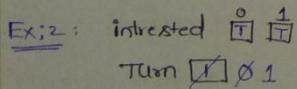
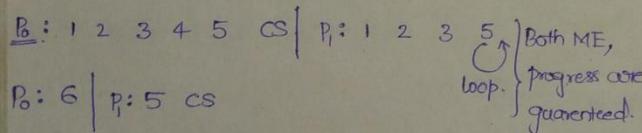
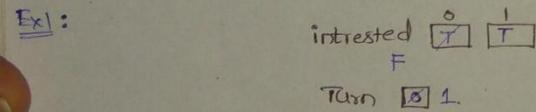
```

14. TRACING PETERSONS SOLUTION

Now Every process should go across Entry section → critical section → Exit section

Now P₀, P₁ are the 2 processes

Ex1:



One thing clear here is the process that executes line no. 4 has the chance to enter/will enter the CS first.

⇒ Generic solution → platform independent.

(2)

⇒ The solution solve the

→ Sleep means
will go
its user
sleep or

16. SLEEP

sleep() c

then it w
wake up

⇒ The mos
problem

→ produces
and th
consum
has o

⇒ Both p

SLEEPING SECTIONS

```

#define N 2
#define TRUE 1
#define FALSE 0
int interested[N] = FALSE;
int turn;
void Entry-Section(int process)
{
    1) int other;
    2) other = !process;
    3) interested[process] = TRUE;
    4) turn = process;
    5) while (interested[other] == TRUE && TURN == process);
}

void Exit-Section(int process)
{
    6) interested[process] = FALSE;
}

```

Void Exit-Section(int process)

```

    6) interested[process] = FALSE;
}

```

15. SYNCHRONISATION MECHANISMS WITHOUT BUSY WAITING

→ The solutions that we have seen till now are subjected to Busy waiting, to solve this we use sleep and wakeup mechanism.

→ Sleep means if any process wants CS and if it is not available that process will go and sleep and as soon as the other process which is using CS finishes its usage it is going to go and wakeup which is sleeping. So using sleep and wakeup we can solve the problem of CPU wastage.

16. SLEEP AND WAKE

Sleep() and wake() are two system calls. If any process calls sleep() then it will go and get blocked and if any process calls wake() it will wake up one of the blocked process.

→ The most common application of sleep and wake is producer and consumer problem.

→ producer means (if there are 2 processes & one process is producing something and the other process is consuming something then it is called producer consumer problem. One process writes & the other process reads what other has written).

→ Both producer and consumer will be using buffers.

```

#define N 100 // slots in Buffer
#define count = 0 // items in Buffer
void consumer(void)
{
    int item;
    while (TRUE) {
        if (Count == 0) sleep();
        item = remove-item();
        Count = Count - 1;
        if (Count == N-1) wakeup(producer);
    }
}

void producer(void)
{
    int temp;
    while (TRUE) {
        item = produce-item();
        if (Count == N) sleep();
        insert-item(item);
        Count = Count + 1;
        if (Count == 1) wakeup(consumer);
    }
}

```

Some gate questions are when is each one going to wakeup others

The problem with producer consumer problem is if the consumer sleeps (count == 0) then it will go to sleep, assume when the consumer is about to sleep it got preempted, now the producer produces/adds item to the Buffer and it will increase count from the value '0' and try to wake up the consumer thinking that the consumer is sleeping, Now whenever the consumer (after preemption) gets scheduled he will goto sleep(), and when the Buffers are full the producer thinks that the consumer will wakeup^{him} and producer also sleeps. ultimately both end up in sleeping forever. so we need to have "Integer" that counts the no. of wakeups(sleeps) and if any process wakes up it will decrement the variable value and the variable is called Semaphore.

7. INTRODUCTION TO SEMAPHORES

- ⇒ Semaphore concept was proposed by DIJKSTRA in 1965
- ⇒ Implementing Semaphore at the user level will be dangerous & inconsistent.
- ⇒ Variables on which Read, Modify, and update happens atomically in Kernel mode (NO preemption)
- ⇒ They are two types
 1. counting semaphore
 2. Binary Semaphore (Mutexes)

18. COUNTING SEMAPHORES

The counting semaphore is a structure like this

struct Semaphore

{

 int value; → Denotes the no. of processes that can access the CS at the

 Queue type L; → Same time some CS are/can be accessed by many processes.

 Down (Semaphore s)

 { S.value = S.value - 1;
 if (S.value < 0)
 { Represent the set of processes that are blocked whenever
 they try to enter the critical section because they
 don't have permission.

 put process (pcb) in L;

 sleep();

 else

 return;

}

 up (Semaphore s)

 { S.value = S.value + 1;

 if (S.value <= 0)

 { select a process from L; wakeup();

Entry Se

Exit Se

→ when a process statements and Incr is less than under star in waiting wake the

⇒ To get one pr

19. PROBLEMS

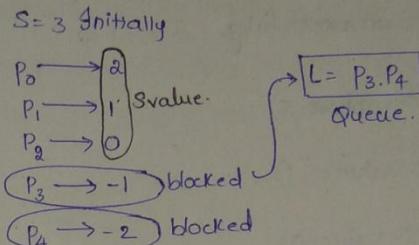
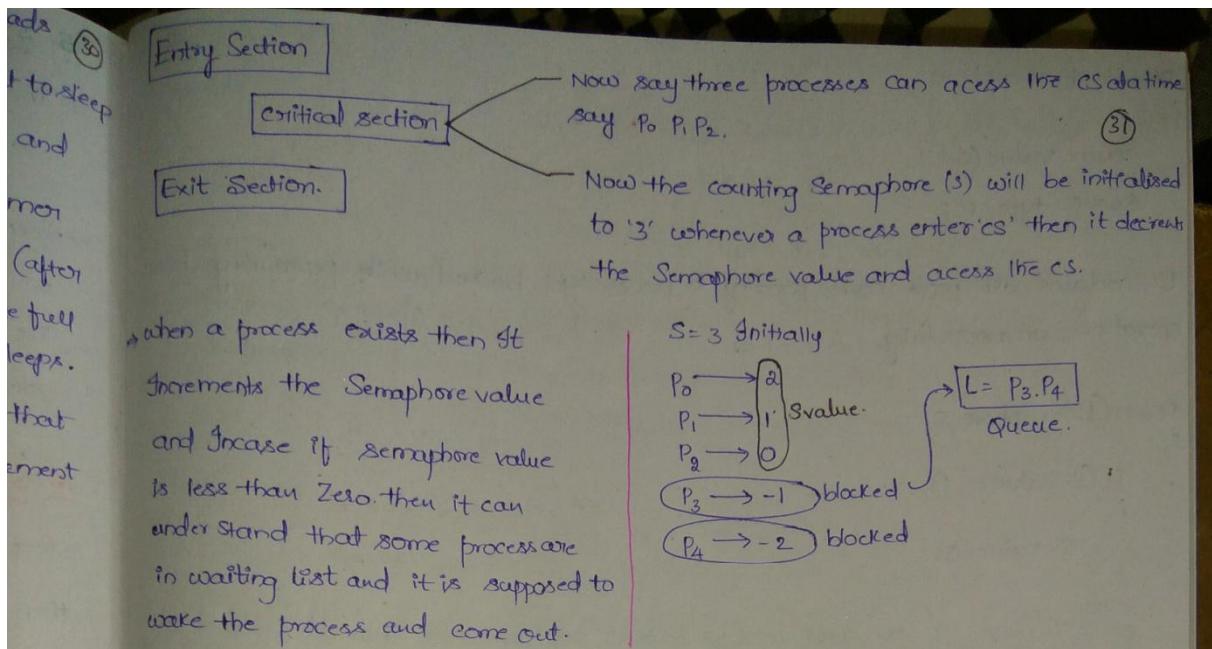
A counting operation

S=10,

20. BINARY SEMAPHORES

→ can have

→ The Bin



⇒ To get mutual Exclusion set the initial value of Semaphore = 1 then only one process can access the CS at a time.

Decrement	Increment
Down	Up
P	V
wait	signal

19. PROBLEMS ON COUNTING SEMAPHORE

A counting semaphore was initialised to 10. The 6P(wait) and 4V signal operations are computed on this semaphore. What is the result

$$\begin{aligned}
 S &= 10, \quad 6P \text{ and } 4V \\
 &\Rightarrow 10 - 6 + 4 \\
 &= 8 //
 \end{aligned}$$

$$\begin{aligned}
 &\text{(2) } S = 7, \text{ then } 20P / 15V \\
 &\Rightarrow 7 - 20 + 15 \\
 &= 2 \\
 &= 2 //
 \end{aligned}$$

20. BINARY SEMAPHORE OR MUTEXES

→ can have only 2 values 0 and 1.

→ The Binary Semaphore is a structure

Struct BSemaphore

```
{  
    enum value(0,1);  
    Queue type L;  
}
```

'L' contains all PCB's corresponding to process got blocked while performing down operation unsuccessfully.

Down(BSemaphore s)

```
{  
    if(s.value == 1)  
    {  
        s.value = 0;  
    }  
    else  
    {  
        put the process (PCB) in S.L;  
        sleep();  
    }  
}
```

Up(BSemaphore s)

```
{  
    if(S.L is empty)  
    {  
        s.value = 1;  
    }  
    else  
    {  
        Select a process from S.L;  
        wake up();  
    }  
}
```

Q1. GATE 92 QUESTION ON MUTEX

mutex = 1

P_i = 10

P_i = 1, 2, ..., 9

while(1)

while(1)

{

down(mutex)

up(mutex)

<CS>

<CS>

up(mutex)

up(mutex)

How many processes can be present in CS at max?

No.
mu

P_i =

whi

{

D

3

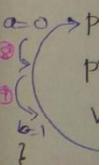
3 M

mates

P₀

while

{



Q3. M

Mute

P₀

while(t)

{

1. pca

2. pcb

{

3. vca

4. vcb

{

(32)

Now,

mutex = 1

 $P_i = 1, 2, \dots, 9$

while (1)

{

arming down

Down (mutex) \Rightarrow mutex = 0.<CS> (P_i) \downarrow
up(mutex)

{ } ①

Now the other process

{ $P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9$ } are

in queue.

25 AUGUST 2017

(33)

 P_{10} .
while (1)

up(mutex) mutex = 1

<CS> P_{10}

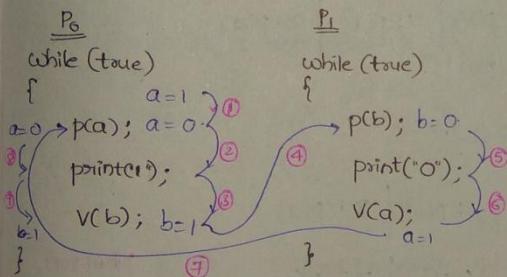
up(mutex)

{ }

Now Initially P_i enters the critical section Now, as P_{10} executes in reverse order (2 ups) it allows all the process one by one to enter the CS. $\therefore \{P_1 - P_9\}$ can be present in CS at any time.22. MUTEX EXAMPLE

mutex a,b; a=1; b=0;

Ans: 10101010-----

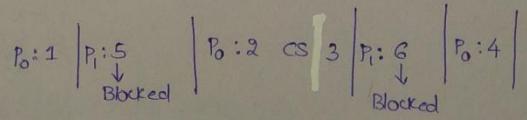
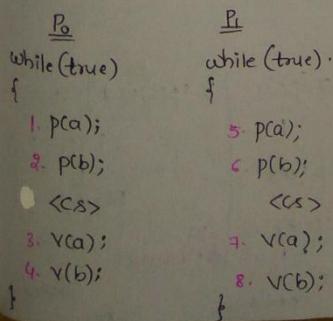


Hence the mutexes are not just used for mutual exclusion but also for the order in which we schedule the processes.

101010-----

23. MUTEX EXAMPLE 1

Mutex a,b; a=1,b=1;



ME is guaranteed.

24. GATE 2013 QUESTION ON MUTEXES

(84)

Three concurrent processes x, y, z execute 3 different code segments that access and update certain shared variables. Process x executes the 'P' operation on semaphores a, b, c and process y executes 'P' operation on semaphores 'b' and 'c' and; process z executes 'P' operation on semaphores c, d, a before entering the respective code segments. After completing the execution of its code segments each process invokes 'V' operation on its 3 semaphores. All are binary semaphores and are initialised to one. Which of the following operations represents deadlock free order of invoking 'P' operation by the processes.

25. 50

Let me
process

a) Thra

Now,

a)

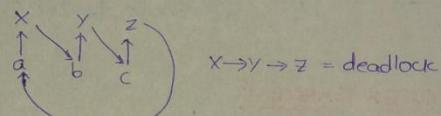
$x: p(a) p(b) p(c)$

$y: p(b) p(c) p(d)$

$z: p(c) p(d) p(a)$

$x: p(a) | y: p(b) | z: p(c)$

$x: p(b) | y: p(c) | z: p(c) p(a)$ DEADLOCK
 ↓ Blocked. ↓ Blocked ↓ Blocked



$x \rightarrow y \rightarrow z = \text{deadlock}$

X

Mutual Ex

$\Rightarrow P_0:$

$P_2:$

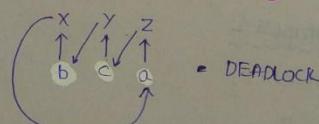
b) $x: p(b) p(a) p(c)$

$y: p(b) p(c) p(d)$

$z: p(a) p(c) p(d)$

$x: p(b) | y: p(b) | z: p(a)$
 $x: p(a) |$ Wait state

$x: p(b) | y: p(c) | z: p(a)$
 $x: p(a) |$ DEADLOCK
 ↓ B ↓ B ↓ B.

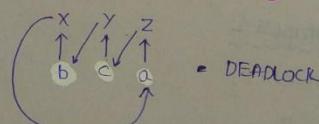


c) $x: p(b) p(a) p(c)$

$y: p(c) p(b) p(d)$

$z: p(a) p(c) p(d)$

$x: p(b) | y: p(c) | z: p(a)$
 $x: p(a) |$ DEADLOCK
 ↓ B ↓ B ↓ B.



d) $x: p(a) p(b) p(c)$

$y: p(c) p(b) p(d)$

$z: p(c) p(d) p(a)$

$x: p(a) | y: p(c) | z: p(c)$
 $x: p(b) | y: p(b) | z: p(d)$
 $x: p(c) | y: p(d) | z: p(a)$ DEADLOCK
 ↓ B ↓ B ↓ B.

$z: p(c) p(d) | x: p(a) p(b)$
 $y: p(c) | z: p(d)$
 ↓ Blocked ↓ Blocked

$z: p(c) p(d) | x: p(c)$
 $y: p(c) | z: p(d)$
 ↓ Blocked ↓ Blocked

26. GATE 9

$P_i : 0, 1, 2, 1$

$M_j : 0, 1, 2, 1$

$P_i : p(m_j)$

<cs>

$v(m_j) : v(n)$

(84)

that access and
on semaphores
dd; process 2
re code segments
'V' operation
led to one.
invoking P operation

25. GATE 2000 QUESTION ON MUTEX

Let $m[0] \dots m[4]$ be mutexes (Binary semaphores) and $p[0] \dots p[4]$ be processes. Suppose each process $p[i]$ executes the following.

$\text{wait}(m[i]); \text{wait}(m[(i+1) \bmod 4]);$ } This could cause
----- Critical Section
 $\text{release}(m[i]); \text{release}(m[(i+1) \bmod 4]);$

- a) Thrashing b) Deadlock c) Starvation d) None of the above
but not DL

Now, $P_0 : P(m[0]) \quad P(m[1]) \Rightarrow P_0$ executes down operation on $m[0]$ and $m[1].$

$P_1 : P(m[1]) \quad P(m[2])$

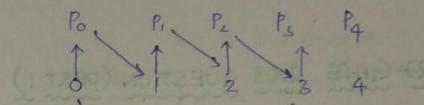
$P_2 : P(m[2]) \quad P(m[3])$

$P_3 : P(m[3]) \quad P(m[0])$

$P_4 : P(m[4]) \quad P(m[1])$

DEADLOCK

deadlock



~~P(m[1])~~ | $P_1 : P(m[2])$ | ~~P(m[2])~~ | $P_2 : P(m[3])$ | $P_3 : P(m[0])$ \therefore Deadlock is possible

Mutual Exclusion is not possible. P_0 and P_2 can enter CS at same time

$\Rightarrow P_0 : P(m[0]) \quad P(m[1]) \Rightarrow CS.$ } P_0, P_2 are operating on 2 different mutexes
 $P_2 : P(m[2]) \quad P(m[3]) \Rightarrow CS$ } so they can enter CS at same time

DEADLOCK

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

These processes can enter the CS at the same time.
so at any time max 2 process can be in CS.

26. GATE 96 QUESTION ON DINING PHILOSOPHER PROBLEM

$x : p(a) p(b)$
 $y : p(c) \downarrow z : p(a)$
 \downarrow
 $x : p(c)$

$P_1 : O_1, \varnothing, I_3$

$M_1 : O_1, 2, I_3$

$P_1 : p(m_1), p(m_{(i+1) \bmod 4});$

$\leftarrow CS$

$v(m_1); v(m_{(i+1) \bmod 4});$

$P_0 : P(m_0) \quad P(m_1)$

$P_1 : p(m_1) \quad p(m_0)$

$P_2 : p(m_2) \quad p(m_3)$

$P_3 : p(m_3) \quad p(m_0)$

This sequence
results in deadlock
we have seen before
Now to avoid DL
let us modify the

Now, after modifying the sequences will be

(36)

$P_0 : p(m_0) \quad p(m_1)$

$P_1 : p(m_1) \quad p(m_2)$

$P_2 : p(m_2) \quad p(m_3)$

$P_3 : p(m_0) \quad p(m_3)$

$P_0 : p(m_0) \quad | \quad P_1 : p(m_1) \quad | \quad P_2 : p(m_2) \quad | \quad P_3 : p(m_0)$

↓
Blocked

$P_0 : p(m_1) \quad | \quad P_1 : p(m_2) \quad | \quad P_2 : p(m_3) \quad | \quad P_3 : p(m_0)$

↓
Blocked

$P_0 : p(m_2) \quad | \quad P_1 : p(m_3) \quad | \quad P_2 : p(m_0) \quad | \quad P_3 : p(m_1)$

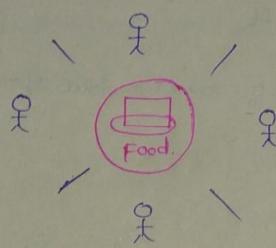
↓
Blocked

$P_0 : p(m_3) \quad | \quad P_1 : p(m_0) \quad | \quad P_2 : p(m_1) \quad | \quad P_3 : p(m_2)$

↓
Blocked

$P_0 : p(m_0) \quad | \quad P_1 : p(m_1) \quad | \quad P_2 : p(m_2) \quad | \quad P_3 : p(m_3)$

↓
Blocked



→ Make one person to take right spoon first and then left spoon

→ Make all others to take left spoon and then right spoon first

→ To break the Deadlock (GATE)

Q:

b) u
a

A) (P

B) (P

C) (P

D) (K

A)

preem

27. GATE 2003 QUESTION (PART 1)

Suppose we want to synchronise 2 concurrent processes 'P' and 'Q' using Binary Semaphores S and T. The code for the process 'P' and 'Q' is shown below.

process P:

while (1) {

W: print '0'

print '0';

X:

}

process Q:

while (1) {

Y: print '1';

print '1';

Z:

Synchronisation points

Can be inserted only at points w,x,y,z

a) which of the following will always lead to an output starting with 001100110011?

A) (P(S) at W) (V(S) at X), (P(T) at Y) (V(T) at Z) (S=1 and T=0) initially 1

B) (P(S) at W), (V(T) at X) (P(T) at Y), (V(S) at Z) (S=1, T=0) initially.

C) (P(S) at W) (V(T) at X) (P(T) at Y), (V(S) at Z) (S=T=1) initially

D) (P(S) at W) (V(S) at X) (P(T) at Y) (V(T) at Z) (S initially 1, T initially 0)

$S=1, T=0 \quad \underline{P}$

W: $p(s) \Rightarrow (S=0)$

print('00');

X: $v(t) \Rightarrow (T=1)$

\underline{Q}

$p(t) \Rightarrow T=0$

print('11');

Z: $v(s) \quad S=1$

∴ Alternate 001100110011-

will be printed.

Q8: GATE 2003 QUESTION (PART 2)

b) which of the following will ensure that the output string never contains a substring of the form 01^m0 or 10^m1 where 'm' is odd.

X 2,3. A) $(P(s) \text{ at } W) (V(s) \text{ at } X) (P(T) \text{ at } Y) (V(T) \text{ at } Z) (S=T=1)$

B) $(P(s) \text{ at } W) (V(T) \text{ at } X) (P(T) \text{ at } Y) (V(s) \text{ at } Z) (S=T=1)$

C) $(P(s) \text{ at } W) (V(s) \text{ at } X) (P(s) \text{ at } Y) (V(s) \text{ at } Z) (S=T=1)$

D) $(V(s) \text{ at } W) (V(T) \text{ at } X) (P(s) \text{ at } Y) (P(T) \text{ at } Z) (S=T=1)$

use right
left spoon

e left spoon
first

GATE)

Binary

w.

Q9: GATE 96 QUESTION ON BARRIER (PART 1)

⇒ Barrier is also one kind of synchronization mechanism.

Barrier is a synchronization construct where a set of processes synchronises globally i.e. each process in the set arrives at the barrier and waits for all others to arrive and then all process leave the barrier. Let the no. of process in the set be three and S be a Binary semaphore with the usual 'P' and 'V' functions. Consider the following 'C' implementation of barrier with line nos shown on left;

Void barrier (void){

1. P(S);
2. process-arrived++;
3. V(S);
4. while (process arrived != 3);
5. P(S);
6. process-left++;
7. if (process-left == 3) {
8. process-arrived = 0;
9. process-left = 0;

10. }

11. V(S);

};

The variables process-arrived, process-left are shared among all process and are initialised to '0'. In a concurrent program all the three process call barrier function when they need to synchronise Globally.

A) The above implementation of barrier is incorrect. Which one of the following is true?

- a) The barrier implementation is wrong due to the use of Binary Semaphore S.
- b) The barrier implementation may lead to deadlock if 2 barrier invocations are used in immediate succession. \hookrightarrow NOT always.
- c) Lines 6 to 10 need not be inside CS.
- d) The barrier implementation is correct if there are only 2 processes instead of three.

\Rightarrow Consider all the process arrived and all the process are leaving out now while leaving all the process that has left first enters the barrier again (because 2 successions are given) then it increments the process arrived but the last process leaving the CS makes process_arrived and process_left = 0 even though a process is executing while loop so, this implementation fails.

30. GATE 08 QUESTION ON BARRIER (PART 2)

How do you fix the above problem?

The main problem is before making process_arrived and process_left = 0 the value is incremented because of the process that has left critical section has arrived again.

The solution is the first process which is reentering should wait until the process arrived = 0

31. GATE 2008 QUESTION ON IMPLEMENTING COUNTING SEMAPHORE USING MUTEX

The 'p' and 'v' operations on counting semaphores, where S is counting semaphore are defined as follows.

p(s): $S = S - 1;$

if (S < 0) then wait;

v(s): $S = S + 1;$

if S < 0 then wake up process waiting on S.

Assume that P_b and V_b the wait and signal operations on Binary semaphores are provided. Two Binary semaphores X_b and Y_b are used to implement the semaphore operations p(s) and v(s) as follows:

$p(s): P_b(X_b);$

$S = S - 1;$

if ($S < 0$)

These cannot be interchanged
 \hookrightarrow $P_b(Y_b)$
(deadlock)

else V_b

\Rightarrow The initial process w/ of A & B

\Rightarrow Now, every \therefore The vo

OR

The counting S

$S = 4$

L []

$\therefore \left\{ \begin{array}{l} \text{let } X_b = 1 \\ \text{L } [] \end{array} \right.$

This contains process that to get access

\Rightarrow Now to g

\Rightarrow Now, let 3 will be in L

following is

(38)

Semaphore S.

Operations are

instead of three

$p(s): P_b(x_b);$

$s = s - 1;$

if ($s < 0$) {

These cannot be interchanged
(deadlock)
} $V_b(x_b);$
 $P_b(y_b);$

else $V_b(x_b);$

$v(s): P_b(x_b);$

$s = s + 1;$

if ($s \geq 0$) $V_b(y_b);$

$V_b(x_b);$

(39)

The initial values of x_b and y_b are respectively?

① 0 and 0 ② 0 and 1 ③ 1 and 0 ④ 1 and 1

goes now

gain because

the last

eventually

⇒ The initial value of x_b cannot be zero because if it is zero all the processes will get blocked. No process can access the critical section.
∴ AIB are eliminated.

⇒ Now, every process that performs down on y_b gets blocked (should be blocked).
∴ The value of y_b should be '0'.

∴ Ans = 1,0.

OR

0 the
station has

still the

IN BY MUX
Semaphore

$S = 4$
L $\boxed{\quad}$

| Now, we should prevent two processes accessing the Semaphore 'S' at the same time. ∴ we use 2 semaphore (mutex_semaphore) to protect the counting Semaphore 'S' and, the list(queue)

∴ $\left\{ \begin{array}{l} \text{let } x_b = \{0, 1\} \\ L_1 \boxed{\quad} \end{array} \right\} \quad \left\{ \begin{array}{l} y_b = \{0, 1\} \\ L_2 \boxed{\quad} \end{array} \right\}$ This contains set of processes that are blocked while accessing 'S'

This contains set of

process that are waiting

to get access to Semaphore 'S'.

⇒ Now to get access to the 4th Semaphore 'S' it should down the x_b and

∴ x_b initially should be 1

semaphores
ment

⇒ Now, Let 4 processes execute the CS then 1 process will be in CS and remain

3 will be in L_1 and P_5 (5th process) try to access the Semaphore it will get blocked

and will be in L_2 ∴ Blocking occurs when we perform down on '0' ∴ $y_b = 0$

32. GATE 2010 QUESTION ON MUTEXES

The following program consists of 3 concurrent processes and 3 binary semaphores.

The semaphores are initialised as $S_0=1, S_1=0$.

(40)

<u>Process P₀</u>	<u>Process P₁</u>	<u>Process P₂</u>
while (true)	wait (S_1)	wait (S_2)
{	Release (S_0);	Release (S_0);
1) wait (S_0)		
2) print "0";		
3) Release (S_1);		
4) Release (S_2);		

How many times P_0 will print 0?

- a) Atleast twice (b) Exactly twice (c) Exactly thrice (d) Exactly once

Given, $S_0=1, S_1=0, S_2=0 \quad S_1 <_1^0 \Rightarrow S_0=1, S_1=1, S_2=0 \quad S_0=1, S_1=0, S_2=0 \quad \left. \begin{array}{l} S_0=1, S_1=1, S_2=0 \\ S_0=1, S_1=0, S_2=0 \end{array} \right\}$ Try with both combinations

$$S_0=1, S_1=0, S_2=0$$

$$P_0: 1 2 3 4 \mid P_1: 1 2$$

$$P_0: 1 2 3 4 \mid P_2: 1 2$$

$$P_0: 1 2 3 4 \mid P_0: 1 2$$

$$P_0: 1 2 3 4 \mid P_1: 1 2$$

$$P_2: 1 2 \mid P_0: 1 2 3 4$$

2 0's printed.

∴ Almost 3 times

(000) \Rightarrow 3 times

$$P_2: 1 2 \mid P_1: 1 2 \mid P_0: 1 2 3 4$$

0 (Con't
printed)

33. GATE 2013 QUESTION ON COUNTING SEMAPHORE

W: $\text{int } x=0$ $x: p(s)$

$p(s)$

$x = x + 1$

$v(s)$

$y: p(s)$

$x = x - 2$

$v(s)$

$z: p(s)$

$x = x - 2$

$v(s)$

'S' is a semaphore whose value is initialised to two! What is the max value of 'x'.

Here the Semaphore is counting Semaphore

(41)

Now, there are only two processes that are incrementing the value of 'x' by $(i+1)$ and $(j+1)$ by 'w' and 'x'!

least value = -4. $\underline{(x-1)} \quad \underline{(x-2)}$ by y, z

34. GATE 1995 QUESTION ON CONCURRENT PROCESS AND SEMAPHORES

Draw the precedence Graph for statements s_1 to s_9 .

var: a,b,c,d,e,f,g,h,i,j,k : Semaphore; All are mutexes & initialised to '0'

begin

cobegin

begin: $s_1; v(a); v(b)$ end;

Enable a,b $\Rightarrow s_2, s_3$ can be executed.

begin: $p(a); s_2; v(c); v(d)$; end;

begin: $p(c); s_4; v(e)$; end;

begin: $p(d); s_5; v(f)$; end;

begin: $p(e); p(f); s_7; v(k)$; end;

begin: $p(b); s_3; v(g); v(h)$; end;

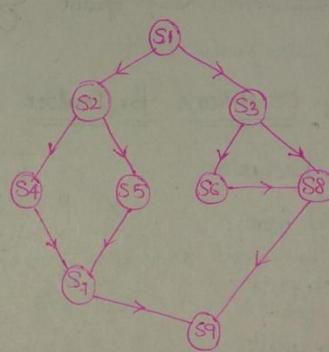
begin: $p(g); s_6; v(i)$; end;

begin: $p(h); p(i); s_8; v(l)$; end;

begin: $p(j); p(k); s_9$ end;

coend;

end.



3. DEADLOCKS

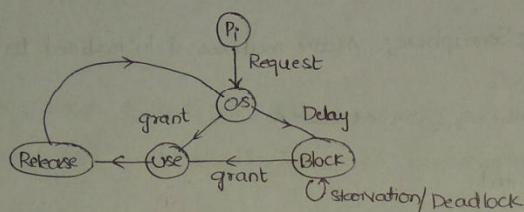
(42)

1. INTRODUCTION TO DEADLOCKS

→ A set of processes are said to be in deadlock, if they wait for happening of an event caused by others in the same set.

→ Starvation is long waiting

→ Deadlock is Infinite waiting



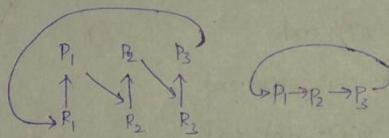
Necessary Conditions for Deadlock

→ Mutual Exclusion

→ Hold and wait:

→ No preemption

→ Circular wait



GATE 1997 QUESTION AND MORE EXAMPLES

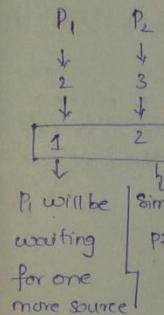
A system is having 3 user processes each requiring 2 units of resource 'R'. The minimum number of units of 'R' such that no deadlock will occur.

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

Let us say

⇒ There are 3 processes and each require 2 units ⇒ $3 \times 2 = 6$ units, so, if there are 6 units of Resource type 'R' then there won't be any deadlock. (But in question they have asked for minimum no. of units.

⇒ Now we might think that 6 units is the minimum. ⇒ P_1 uses 2 resources and releases them. Then P_2 takes resource and releases it and P_3 takes the resource and releases it. But what if P_1 takes one unit of Resource and P_2 takes one unit of Resource then P_1 waits for P_2 and P_2 waits for P_1 so deadlock occurs.



⇒ Now the answer

(42) \Rightarrow Now if I have 3 processes say P_1 - 1 unit, P_2 - 1 unit, P_3 - 1 unit and here P_1 waits for P_2 , P_2 waits for P_3 and P_3 waits for $P_1 \Rightarrow$ DEADLOCK (43)

Now, if I have 4 units it will solve my problem

P_1	P_2	P_3	(4) remaining
1	1	1	

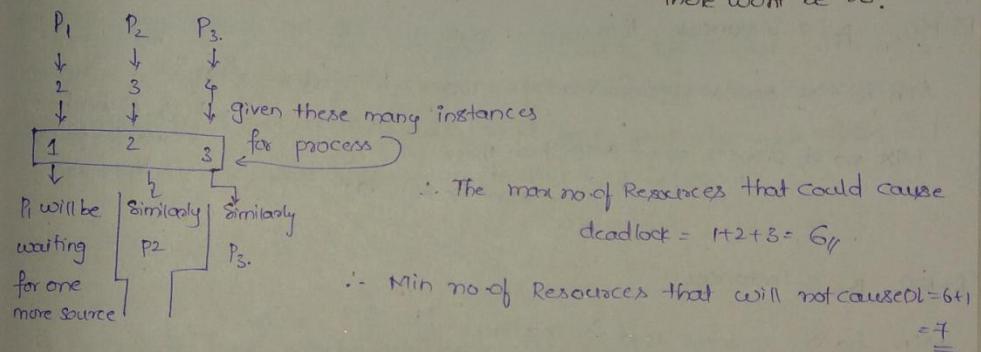
P_1 comes out, P_2 comes out, P_3 comes out

\Rightarrow The max no. of Resource without could cause deadlock = 3.

\Rightarrow The min no. of units reqd so that No deadlock will occur = 4 \Rightarrow Now to find this Find the max no of units that could cause deadlock and add 1 to it.

\Rightarrow Now consider process $P_1 P_2 P_3$ Requires { then how many resources are reqd in minimum so that there wont be DL?

P_1	P_2	P_3	2 3 4 Resources
↓	↓	↓	
2	3	4	



Since 'R' The Let us say $P_1 P_2 P_3 \dots P_n$

$$x_1 x_2 x_3 \dots x_n$$

$$(x_1-1) + (x_2-1) + (x_3-1) + \dots + (x_n-1) = \text{The max no of resources that when allocated can lead to deadlock}$$

$$= (x_1 + x_2 + x_3 + \dots + x_n) - n$$

$$= \left(\sum_{i=1}^n x_i \right) - n \Rightarrow \text{It is the value of max no of resources that could cause deadlock.}$$

\therefore Min no of Resources that cannot/will not cause deadlock

$$= \left(\sum_{i=1}^n x_i - n \right) + 1$$

\Rightarrow Now they might ask the Question in Reverse way a Resource has '6' instances and Each process requires $P_i = 2$ instances what is the max no of pro

Given $R=6$ instances $P_i = 2$ instances.

(4)

Min. no. of process which result in deadlock = $P_1 P_2 P_3 P_4 P_5 P_6$

Instances: $\begin{matrix} | & | & | & | & | & | \\ \downarrow & & & & & \\ \text{wait for} & & & & & \\ \text{one instance} & & & & & \end{matrix}$

\therefore The max no. of process that can operate without deadlock = $5 \times (G-1) = 5 \times (6-1) = 25$

\Rightarrow Total instances.

$R=6$, $P_i = 3$. (Each require 3 instances so give 2 instances for them and make them wait for one more instance)

$P_1 P_2 P_3$. \therefore Then three process results in DL
 $2 2 2$. \therefore No. of process that doesn't cause DL = 2.

$R=100$, $P_i = 2$ instances. Then

Min no. of process reqd for Deadlock = 100. $P_1 P_2 P_3 \dots P_{100}$

$| | | \dots | \Rightarrow$ All will be waiting for one instance

$$\text{Want be DL} = 100 - 1 = \underline{\underline{99}}$$

$R=100$, $P_i = 3$ instances

Min no. of process reqd for Deadlock = $\frac{P_1}{2} \frac{P_2}{2} \dots \frac{P_{50}}{2} = 50$.

Max no. of process reqd such that there won't be DL = 49

$R=100$, $P_i = 4$ instances.

$$m \times (3) = 100$$

$$\Rightarrow m = \lceil \frac{100}{3} \rceil$$

$\boxed{n=34}$ processes

Min no. of process lead to DL = 34

Max no. of process don't lead to DL = 33.

5. GI

Suppo

steser

of

for

a) vi

The

P.

3. GATE 1992 QUESTION

A computer system has 6 tape drives, with n process computing for them. Each process needs 3 tape drives. The max value of n for which the system is guaranteed to be deadlock free

- 2 b) 3 c) 4 d) 1

$$= 5 \times (6 - 1)$$

$\Rightarrow R = 6$, $P_i = \text{No.of instances needed by each process} = 3$.

them and
em wait for
2 instance)

- 2

Now Give 2 instances to each process.

$$\Rightarrow \frac{P_1}{2} = \frac{P_2}{2} = \frac{P_4}{2} \quad \left| \begin{array}{l} \text{Min no of process that can cause deadlock} \\ = 3 \\ \text{Max no of process that results in} \\ \text{DL free} = 2. \end{array} \right.$$

4. GATE 1993 QUESTION ON MINIMUM RESOURCES REQUIRED

will be
ng for one.
-ance

$$m = ? \quad \underline{A} \quad \underline{B} \quad \underline{C}$$

3 4 6

a) 7 b) 9 c) 10 d) ~~13~~

$$\text{Max } 'm' \text{ where there will be deadlock} = 2+3+5 = 10$$

Min no. of Resources where there will not be deadlock = 11. (anything > 11)

5. GATE 2005 QUESTION

Suppose n processes $P_1 - P_n$ share ' m ' identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process P_i is S_i where $S_i > 0$. Which one of the following is a sufficient for ensuring that deadlock does not occur.

- $$a) \forall i, s_i < m \quad (b) \forall i, s_i < n \quad (c) \sum_{i=1}^m s_i < (m+n) \quad (d) \sum_{i=1}^n s_i < (m \times n).$$

There are m Resource units, n -process P_1, P_2, \dots, P_n

P_i - S_i units of Resource

$P_2 \leftarrow S_2$

b - c = n → c units of Resource

$$P_2 - S_2 = \left(\sum_{i=1}^n S_i - n \right), \quad m \leq \sum_{i=1}^n S_i - n \Rightarrow \sum_{i=1}^n S_i < (m+n)$$

6. GATE 06 QUESTION ABOUT NECESSARY CONDITION FOR DEADLOCK

Consider the following snapshot of a system running 'n' process. process 'i' is holding x_i instances of a Resource 'R' for $1 \leq i \leq n$. currently all instances of R are occupied. further, for all 'i' process 'i' has placed a request for an additional y_i instances it already has. There are exactly two process 'p' and 'q' such that $y_p = y_q = 0$. which of the following can serve as necessary condition to guarantee that the system is not approaching a deadlock?

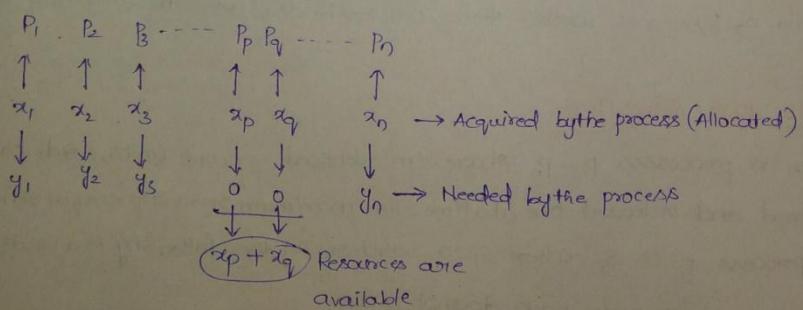
- a) $\min(x_p, x_q) < \max_{k \neq p, q} y_k$
- b) $x_p + x_q \geq \min_{k \neq p, q} y_k$
- c) $\max(x_p, x_q) > 1$
- d) $\min(x_p, x_q) > 1$

R=9

P ₁	P ₂	P ₃	
5	4	6	⇒ Requested
3	4	2	⇒ allocated
2	0	4	= Needed

4 Resources are released now the necessary condition to break the Deadlock is the freed Resource should serve atleast one of the process that is in need of Resource then the Deadlock will be Broken.

Now, Generalising the above concept



$(x_p + x_q)$ should satisfy atleast one of the Remaining process need (atleast it should satisfy the min need of the among the Remaining process).

$$\therefore x_p + x_q \geq \min_{k \neq p, q} y_k$$

7. DEADLOCK

Strate

17 Dead

27 Dead

37 Dead

47 Dead

8. DEADLOCK

⇒ ME

⇒ Hold

⇒ Wait

⇒ Post

⇒ Dis

an

co

no

9. SAFE

At any

E-(6 3 4 2)

P-(5 3 2 2)

↑(6 2 0)

"(1 0 2 0)

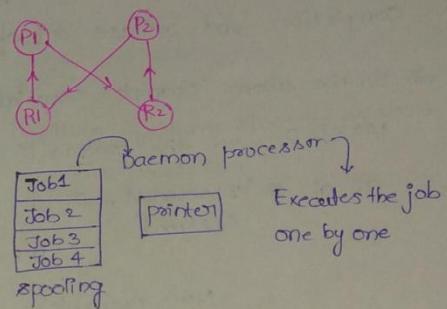
7. DEADLOCK HANDLING MECHANISMS

Strategies for Handling Deadlock

- 1) Deadlock Ignorance - Both windows and Linux uses Deadlock Ignorance
- 2) Deadlock prevention - Disable one of the four necessary conditions.
- 3) Deadlock Avoidance
- 4) Deadlock Detection and Recovery - Most practical method.

8. DEADLOCK PREVENTION

Condition	Approach
Mutual Exclusion	Spool Everything
Hold and wait	Request all resources initially
No preemption	Take resources away
Circular wait	order resources numerically



⇒ ME and Hold and wait disabling are not possible practically.
 ⇒ Hold and wait disabling is not possible because a process cannot declare its need well advanced of the execution.

⇒ preemption causes inconsistency
 ⇒ Disable circular wait is possible (Number all the Resources Initially and now a process is supposed to ask for resources in the increasing order) (P₁ asks R₂ then it should ask for resource R₃, R₄... R_n (n>2) but not R₁) (Can be implemented practically).

9. SAFE, UNSAFE, DEADLOCK AVOIDANCE AND BANKERS ALGORITHM

At any instant of time the snapshot of the system is

	Tape drivers	plotters	scanners	CD-Roms
Remaining	E(6 3 4 2)	A 3	0 1	1 1
Initial	I(5 3 2 2)	B 0	1 0	0 0
Allocation	C 1	1 1	1 0	0 0
Available	D 1	1 0	0 1	1 0
	E 0	0 0	0 0	0 0
	Resources Assigned			

Process	Tape drives	plotters	scanners	CD-Roms
	A 1	1 0	0 0	
	B 0	1 1	1 2	
	C 3	1 0	0 0	
	D 0	0 1	1 0	
	E 2	1 1	0 0	
	Resources still needed			

E = It is a vector that represents the total no. of resources available in system

(18)

$E(6 \ 3 \ 4 \ 2)$
 Tapedrivers | CD Rom's.
 plotters | scanners

⇒ The a
Banker

10. GATE

⇒ Three me
There are
consider

	X
P0	1
P1	2
P2	2

Total = $(5 \ 5)$

Allocated $(5 \ 4)$

Available $(0 \ 1)$

Now, the

present

P, Res

Release

Now,

⇒ P and (A) → How many resource Instances are still available $(E - P)$

↳ How many resources are assigned currently

→ Safe state is a state in which you could satisfy the need in some order

→ whenever the Available satisfies the need of a process it will run till completion and release all of its "Resources assigned."

⇒ In the above example Available = $(10 \ 20)$ if satisfied the need of process 'D' so the process 'D' completes the execution and it releases all of its "resources." $\Rightarrow A = 1 \ 0 \ 2 \ 0$

$$\begin{array}{r} \text{Assigned} \\ A = 1 \ 0 \ 2 \ 0 \\ D = 1 \ 1 \ 0 \ 1 \\ \hline \end{array} \quad \text{'D' satisfied}$$

$$\text{New Available} = \underline{\underline{2 \ 1 \ 2 \ 1}}$$

⇒ with the New Available A can be satisfied

$$\begin{array}{r} \text{New Available} = 2 \ 1 \ 2 \ 1 \\ \underline{\underline{3 \ 0 \ 1 \ 1}} \\ 5 \ 1 \ 3 \ 2 \\ \hline \end{array} \quad \text{'A' satisfied}$$

⇒ with New Available B can be satisfied

$$\begin{array}{r} A = 5 \ 1 \ 3 \ 2 \\ B = \underline{\underline{0 \ 1 \ 0 \ 0}} \\ 5 \ 2 \ 3 \ 2 \\ \hline \end{array} \quad \text{'B' satisfied}$$

⇒ with New Available A = 5 2 3 2

$$\begin{array}{r} \text{'C' can be satisfied} \\ A = 5 \ 2 \ 3 \ 2 \\ C = \underline{\underline{1 \ 1 \ 1 \ 0}} \\ 6 \ 3 \ 4 \ 2 \\ \hline \end{array} \quad \text{'C' satisfied}$$

⇒ If you could satisfy the needs of all the processes in some sequence then it is called safe state.

DABCE = safe state

(48) \Rightarrow The above phase is called Deadlock Avoidance and the Algo is called Banker's Algorithm.

(49)

10. GATE 2007 QUESTION ON SAFE STATE

Three resource types X, Y, Z

There are 5 units of each resource type

consider the following table and say which process finishes last.

me order

until

of

releases

id

=

	X	Y	Z	X	Y	Z
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

Alloc

Request

A) P0

B) P1

C) P2

D) None of above since the system is in Deadlock.

$$\text{Total} = \begin{pmatrix} X & Y & Z \\ 5 & 5 & 5 \end{pmatrix}$$

$$\text{Allocated} = \begin{pmatrix} 5 & 4 & 3 \end{pmatrix}$$

$$\text{Available} = \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}$$

Now, the Available satisfies the need of P1 \Rightarrow after execution P1 releases the resources

$$\text{present available} = \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}$$

$$P_1 \text{ Resource are} = \begin{pmatrix} 2 & 0 & 1 \end{pmatrix}$$

$$\text{Released} \quad \underline{\begin{pmatrix} 2 & 1 & 3 \end{pmatrix}} = \text{New Available}$$

Now, $(2 \ 1 \ 3)$ will satisfy the need of P0 $(1 \ 0 \ 3)$

$$\therefore \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \\ \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \\ \underline{\begin{pmatrix} 3 & 3 & 4 \end{pmatrix}} = \text{New available}$$

Now, New available satisfies the need of P2 $\Rightarrow (3 \ 3 \ 4) \bullet (0 \ 1 \ 2)$

$$\therefore \text{New available} = (3 \ 3 \ 4)$$

$$\text{Total: } \underline{\begin{pmatrix} 2 & 0 & 1 \end{pmatrix}} = \begin{pmatrix} 5 & 5 & 5 \end{pmatrix}$$

$\boxed{P_1 \ P_0 \ P_2}$ is the safe sequence

QUESTION ON SAFE STATE

(5)

12. 5

	Allocated	Maximum	Future need
P _A	1 0 2 1 1	1 1 2 1 3	0 1 0 0 2
P _B	2 0 1 1 0	2 2 2 1 0	0 2 1 0 0
P _C	1 1 0 1 1	2 1 3 1 1	1 0 3 0 0
P _D	1 1 1 1 0	1 1 2 2 0	0 0 1 1 0

Available = (0 0 x 1 1) (Q) what is the smallest value of 'x' for which this is a safe state?

Need = (max - Allocation) \Rightarrow The future need matrix will be

\Rightarrow Using the available we should satisfy any one of the process future need

Now the available (0 0 x 1 1) satisfies need of P_D if x=1

$$= (0 0 1 1 1)$$

$$\therefore \text{Available} = (0 0 1 1 1)$$

$$\begin{array}{r} \text{P}_D \text{ Resources} = (1 1 1 1 0) \\ \hline (1 1 2 2 1) = \text{New Available} \end{array}$$

Now, New Available (1 1 2 2 1) doesn't satisfy any of the process need.

If we observe we have put x=1 but if we put x=2 then the new available will be

$$\text{Avail}(0 0 2 1 1)$$

$$\begin{array}{r} \text{P}_D. (1 1 1 1 0) \\ \hline \end{array}$$

$$\text{NewAvail} = (1 1 3 2 1) \rightarrow \text{This solves/ leaves the need of all other processes.} \therefore \text{for } x=2$$

safe state exists

$\therefore \boxed{\text{Min value of } x=2} \therefore \text{No deadlock}$

Av
Req
Req

Now,
look at
Alloc
X
P₀ 0
P₁ 3
P₂ 2

Here
get

simil

13. GP
A
X
P₀ 1
P₁ 0
P₂ 1

12. GATE 2014 QUESTION ON BANKERS ALGORITHM

	X	Y	Z	X	Y	Z	FUTURE NEED
P ₀	0	0	1	8	4	3	8 4 2
P ₁	3	2	0	6	2	0	3 0 0
P ₂	2	1	1	3	3	3	1 2 2

Allocation Max Need.

Available (3, 2, 2) Now given the system is in safe state then will the Req1,

Req1: P₀(0, 0, 2) Req2 be granted?

Req2: P₁(2, 0, 0). Req1: P₀(0, 0, 2)

Available: (3 2 2)

e need

Now, P₀ needs (0, 0, 2) Now assume i allocated it then, Allocation Need matrices look as, when i allocate it the allocated Resources gets increased and Need of the process gets decreased

Allocation	Need
X Y Z	X Y Z
P ₀ 0 0 3	P ₀ 8 4 0
P ₁ 3 2 0	P ₁ 3 0 0
P ₂ 2 1 1	P ₂ 1 2 2

Available will become (3, 2, 0).

satis by P₁

$\Rightarrow 320 \rightarrow$ New Available

$\frac{320}{320} \rightarrow P_1$ Release resources.

$\frac{320}{640} \rightarrow$ New Available

need.
Deadlock
for x=1

Here the need of P₁ is satisfied, P₀, P₂ does not get satisfied so the Req1 will not be granted.

Similarly perform with P₂ then you will get a safe sequence

13. GATE 1996 ON BANKERS ALGORITHM

Allocation	Max Need			Future Need			Available (2 2 0)
	R ₀	R ₁	R ₂	R ₀	R ₁	R ₂	
P ₀ 1 0 2	4	1	2	3	1	0	Request P ₀ : (0 1 0)
P ₁ 0 3 1	1	5	1	1	2	0	which will the Req be granted
P ₂ 1 0 2	1	2	3	0	2	1	

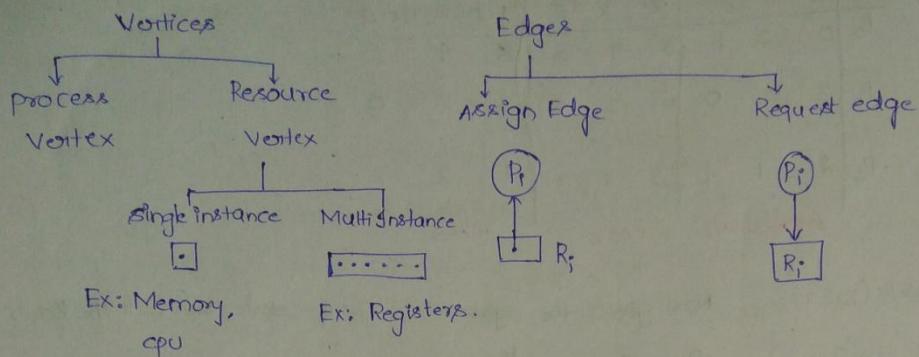
Now, assume i allocated the resource then Allocated, Future Need, Available are going to change.

Allocated Need(Future)

P₀: (1 1 2) P₀: (3 0 0)

Available: (2 1 0) \Rightarrow cannot satisfy Deadlock

4. RESOURCE ALLOCATION GRAPH



I. NEED FOR
→ CPU CO

Registers

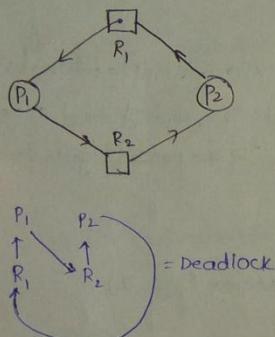
Now, clas

size 4

the CP

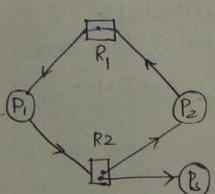
C

5. RESOURCE ALLOCATION GRAPH EXAMPLES



⇒ In case of single instance types if there is a cycle then there will be deadlock

⇒ A presence of cycle in single instance Resource type is sufficient for Deadlock



⇒ In multi instance of Graph the presence of cycle is not the sufficient condition but necessary condition

	Alloc		Request	
	R1	R2	R1	R2
P1	1	0	0	1
P2	0	1	1	0
B	0	1	0	0

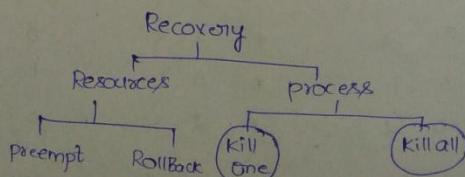
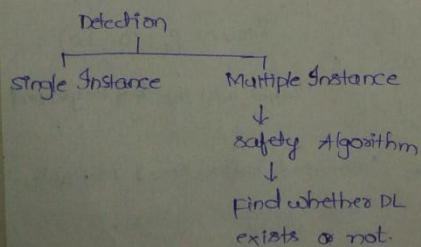
Available = (0 0)

No deadlock, safe sequence exists.

of 'P' + time
time?

⇒

6. DEADLOCK DETECTION AND RECOVERY

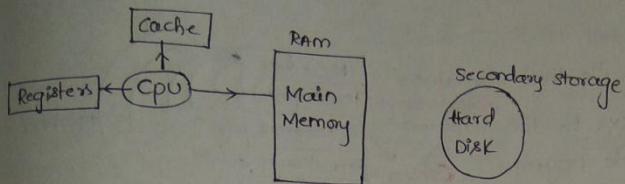


∴ CP

4. MEMORY MANAGEMENT

NEED FOR MULTIPROGRAMMING AND MEMORY MANAGEMENT

→ CPU can directly access Registers, Main memory, Cache (if available)



Now assume the main memory is 4MB and I want to run process of size 4MB but the process spends fraction of 'p' time in doing I/O then the CPU utilisation = ?

CPU utilisation = The amount of time that you are using CPU running the program.

$$\frac{\text{MM}}{4\text{MB}} \quad \frac{\text{Process size}}{4\text{MB}}$$

$$\boxed{\text{CPU utilisation} = (1-p)} = 20\% \quad (\text{80\% time it spends in I/O (assuming)})$$

Now, MM = 16MB process size = 4MB. Each process spends fraction of 'p' time doing I/O what is the prob that all the process do I/O at same time?

$$\Rightarrow \frac{\text{MM}}{4\text{MB}} = 4 \text{ processes} \quad \therefore \text{Now prob that all process do I/O at same time} = p \times p \times p \times p = p^4$$

$$2. \boxed{\text{CPU utilisation} = 1 - p^4} \approx 60\% \quad (\text{assuming 80\% of time spends in I/O})$$

$$\Rightarrow \text{MM} = 32\text{MB} \quad \text{ps} = 4\text{MB} \Rightarrow 8 \text{ processes}$$

$$\boxed{\text{CPU utilisation} = 1 - p^8} \approx 83\%$$

$$\therefore \text{CPU utilisation} = 1 - p^n$$

$$= 1 - \left(\frac{\text{fraction of memory spent in I/O}}{\text{size of memory}} \right)^n$$

Degree of multiprogramming = The no. of processes that can be present in main memory at any given instant of time.

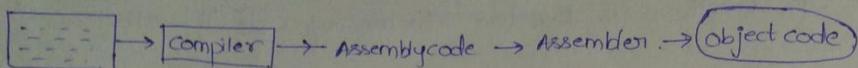
$$\boxed{\text{CPU utilisation} = 1 - p^n} \quad (n = \text{no. of processes in MM}).$$

2. OBJECT CODE, RELOCATION AND LINKER

(54)

We write the program in the Text editor and the further process is

Text editor



Now, the object code is a file that contains
The Header contains pointers to the various
Segments of the object code program.

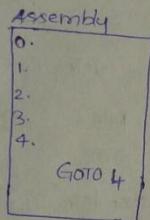
Instructions
Constants
Relocation info

Header
Text Segment
Data Segment
Relocation Info
Symbol Table
Debug Info

pointers.

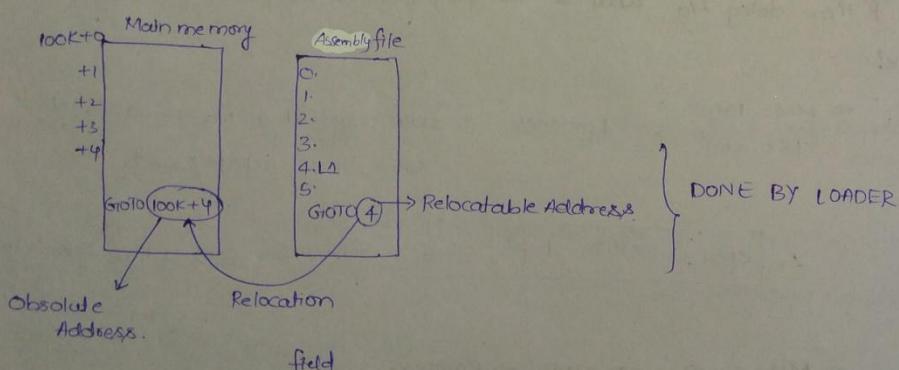
Now this object code file contains "Relocation info". Now the compiler/assembler assume that the line no's starts with Address '0'.

Now,



This works fine when the address starts from '0'
but there is no guarantee that the file (object file) will be loaded ^{into} Main memory from address '0' (All process will not start from address '0').

Now, In the main memory say the object file loaded starting from one address as 100K+0 then,



Now, the "Relocation Info" in the object file contains what are the lines that are changed to (Relocatable Address → absolute Address).

→ Symbol table contains every symbol that is present in the program.

min()	100	Address
point()	-	
scanf()	-	

Unresolved Symbols (their address cannot be found)

Libraries
C
D
E
:

This contains
Relocatable
Addresses bcoz
linker doesn't
know where the
program is loc

phase (like
contain th
so we wr
⇒ The mai

3. LOADER

Object linked

- 1) program load
- 2) Relocation
- 3) Symbol Reso

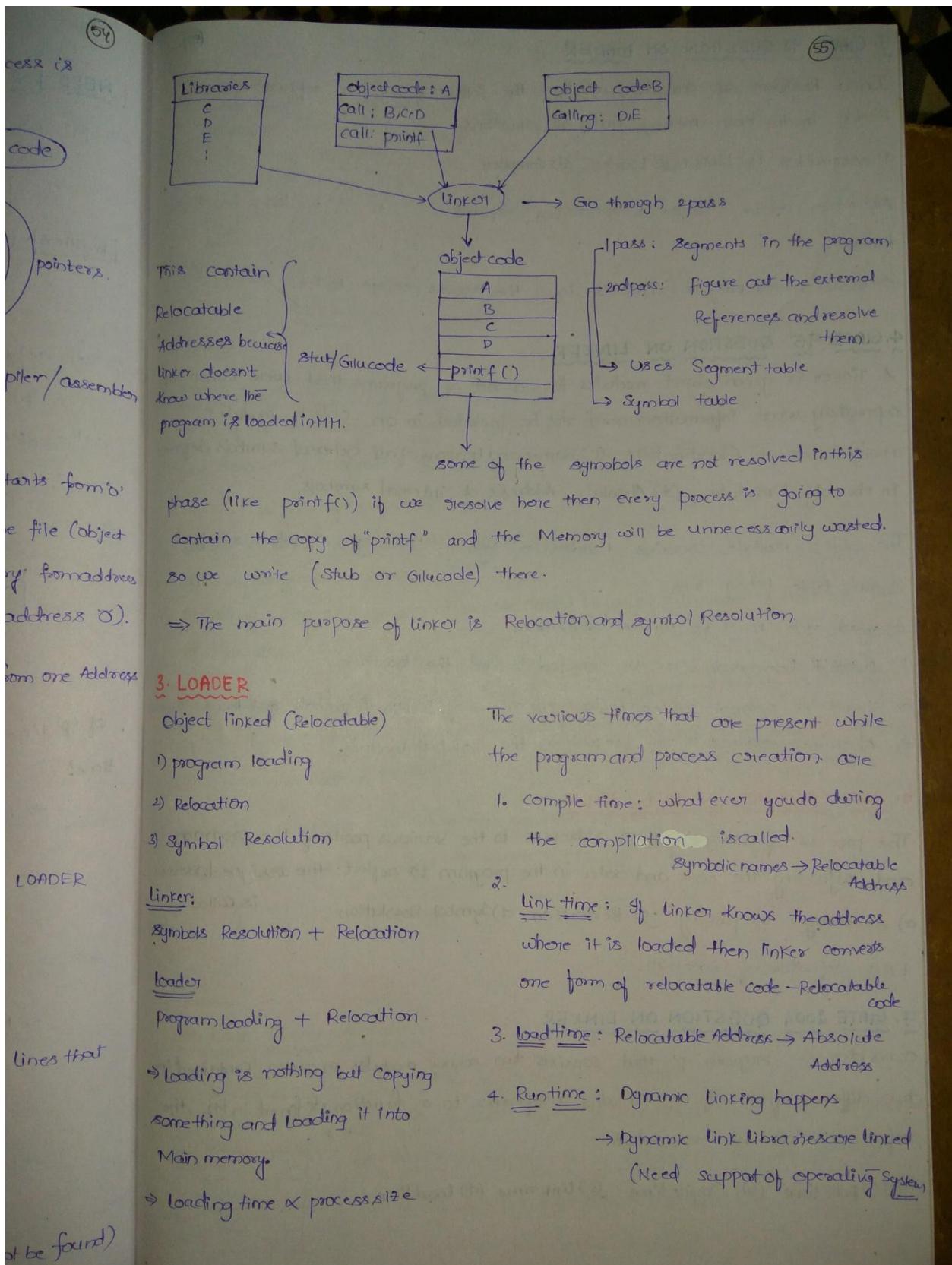
Linker:

Symbols Resolution

Loader:

Program loading

→ Loading is no
something and
Main memory.
⇒ Loading time



5. GATE 98 QUESTION ON LOADER

In a Resident OS computer, which of the following system software must reside in the main memory under all situations.

- (a) Assembler (b) Linker (c) Loader (d) Compiler

Assembler can be loaded whenever we require

Linker " " " " " "

⇒ There is No program that can load the process except loader.

4. GATE 95 QUESTION ON LINKER

A linker is given object modules for a set of programs that were compiled separately. What information need not be included in an object module?

- (a) object code (b) Relocation bits (c) Names and locations of all external symbols defined in the object module (d) Absolute Address of internal symbols.

The object module contains Header, Text Segment, Data Segment, Relocation info, Symbol-table, Debug Info.

- a) object code must be present definitely.
- b) ~~Absolute~~ Relocation bits are needed to find the location.
- c) stored in symbol Tables (Defines the names of External symbols and their address)
- d) Absolute Address is not present in the Object module.

6. 2001 QUESTION ON RELOCATION

The process of assigning load addresses to the various parts of the program and adjusting the code and data in the program to reflect the assigned address

- a) Assembly b) Parsing c) Relocation d) Symbol Resolution

is called —

BASIC definition of Relocation.

7. GATE 2004 QUESTION ON LINKER

Consider a program 'p' that consists two source modules m_1 and m_2 contained in two different files. If m_1 contains a reference to a function defined in M_2 , the reference will be resolved at

- (a) Edit time (b) Compile time (c) Link time (d) Load time

9. GATE

Which

linked

a) Sm

b) Le

c) Fo

d) Ex

of

of

to

b) → p

and

c,d) →

10. FI

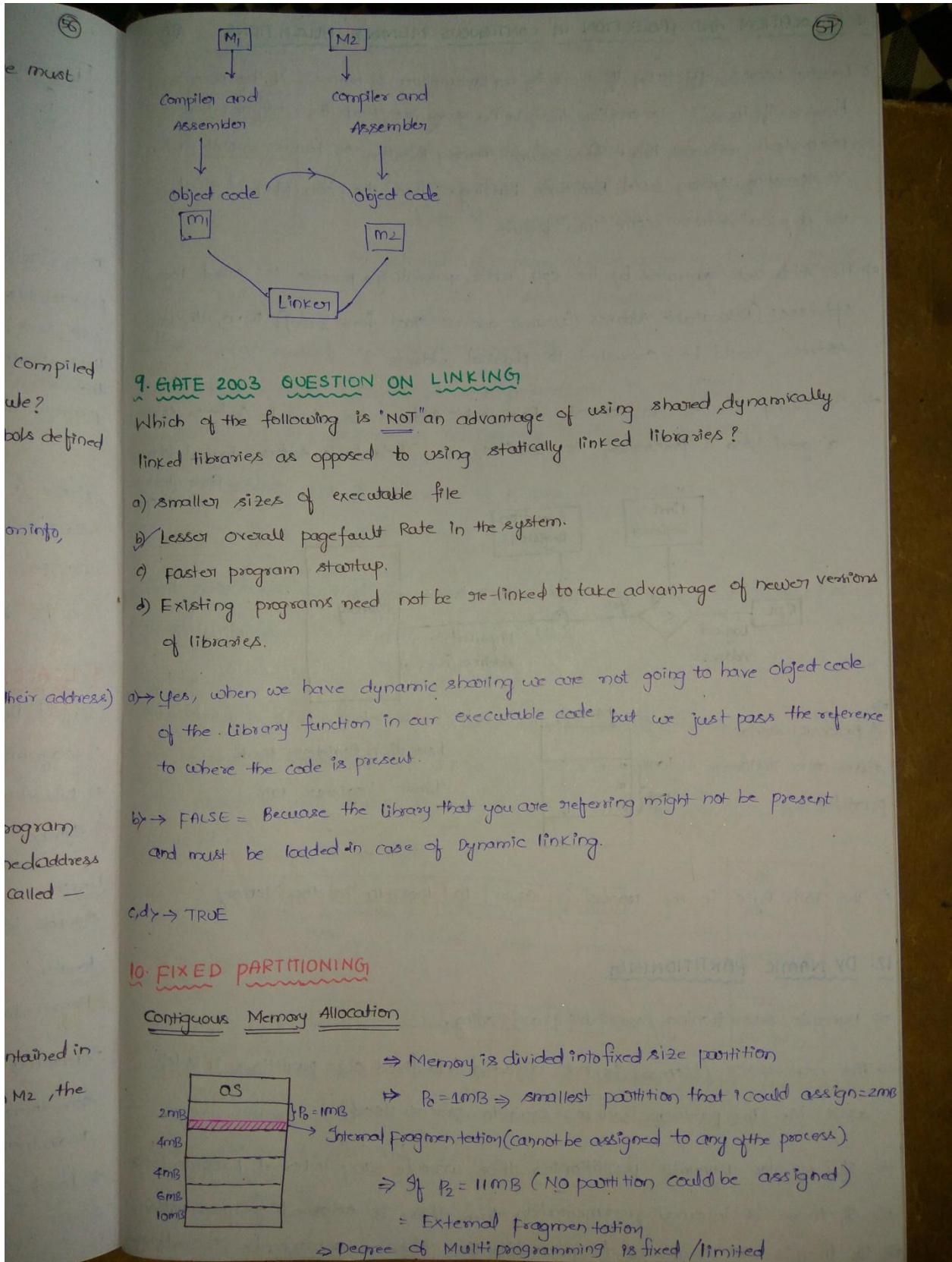
Conti

2MB

4MB

6MB

10MB



II. RELOCATION AND PROTECTION IN CONTIGUOUS MEMORY ALLOCATIONS

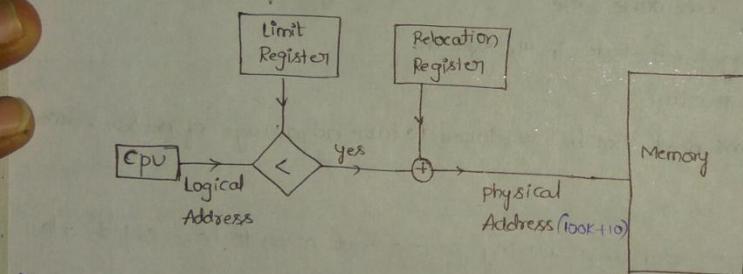
58

⇒ Loader works efficiently if there is no preemption of process in the memory because if there is preemption there is no guarantee that the program that is preempted will not load from same memory location. So Loader doesn't work. So operating system used runtime binding. Thus the concept of logical and physical address come into picture.

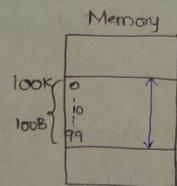


⇒ The addresses generated by the CPU while generating a process is called logical addresses. (Relocatable Address (assume address start from Zero)). Now, the logical address should be converted to physical address.

⇒ Involving loader to convert Relocatable Address to Absolute Address is not a good idea (Preemption is possible). Relocatable Address = Logical Address + Generated by CPU



⇒ A process cannot span over two partitions.



Relocation Register = 100K

Limit Register = 100.

⇒ The 10th Byte in the process is equal to look+10 in the Memory

12. DYNAMIC PARTITIONING

⇒ Dynamic partitioning / variable partitioning.

⇒ The user space / Memory is not divided into fixed size partitions, it is left as it is. (The partition will be equal to process need)

⇒ If i go for Dynamic partitioning there won't be any internal fragmentation.

⇒ If there is internal fragmentation then there is external fragmentation.

⇒ If there is no internal fragmentation then there may or may not be external fragmentation.

⇒ The adva

a) It is

b) Size

⇒ The Allocat

13. BIT MAP

⇒ The Data stru

⇒ The Data stru

Bitmap:

⇒ The memory is

⇒ For every A
the allocati
occupied.

⇒ Now, suppose

in the Bitr

occupied a

of memory

⇒ Bitmap is

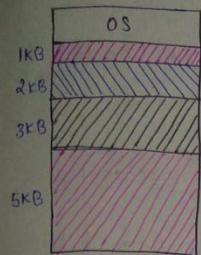
memory
that is
not work.

gical

of logical

the logical

ress is not
ical Address,
generated by
cpu.



⇒ Suppose that all the memory fragments are filled completely ⇒ There is no internal fragmentation. Now if the cells 1KB and 3KB got released but we cannot allocate the space to program of 4 KB because the memory that is freed is not contiguous locations.
∴ External Fragmentation is present.

⇒ Compaction/ Defragmentation ⇒ costly (Bring all empty spaces together)

⇒ The advantages of Dynamic partitions is

- a) It is flexible (Degree of Multiprogramming is not fixed)
- b) size of the process is not limited by the size of partition.

⇒ The Allocation and deallocation of memory is complex.

3. BITMAP FOR DYNAMIC PROGRAMMING

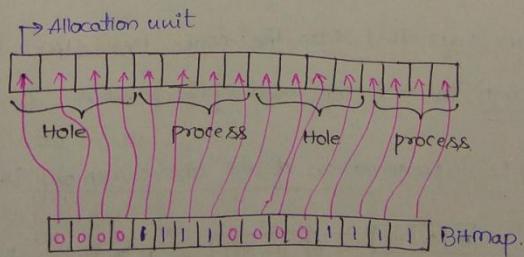
⇒ The Data structures that are used to keep track of free and occupied spaces

⇒ The Datastructure is Bitmap and Linked lists.

Bitmap:

⇒ The memory is divided into small parts called "Allocation units".

⇒ For every Allocation unit the single bit 0 or 1 is assigned (0 represents the allocation unit is open/free and 1 represents the allocation Bitunit is occupied).



OS
Hole
process p1
Hole
process p2

it is left

⇒ Now suppose the size of allocation unit is 4 Bytes = 32 Bits. Now 1bit in the Bitmap is used to represent whether these 4B (Allocation unit) is occupied or not. so the Bitmap is going to take $\frac{1}{(32+1)} = \frac{1}{33}$ rd part of memory is taken by Bitmap.

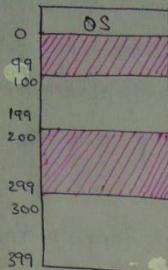
fragmentation.

⇒ Bitmap is not widely used / Not used nowadays.

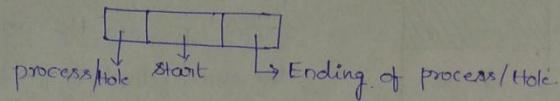
14. LINKED LIST FOR DYNAMIC PARTITIONING

(6)

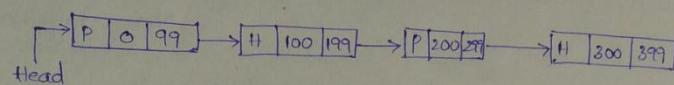
Now, after a



⇒ The structure of the linked list will be like this.



⇒ For the above diagram the structure of linked list will be



⇒ Searching will not take lot of time

⇒ Maintaining nodes in the increasing order of starting Address

⇒ Maintaining as double linked list has advantage, Now whenever you free a process then check if the previous node is hole and then merge them and check the node after hole then you should also merge it. precisely after freeing the process if you get run of holes then merge them into single node

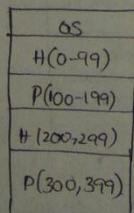
15. FIRST FIT, NEXT FIT, BEST FIT, WORST FIT

The various algorithms that are used on the linked list are

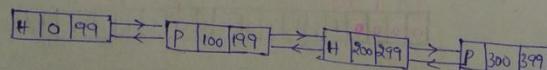
- 1) First fit 3) Best fit
- 2) Next fit 4) Worst fit

First Fit

⇒ The first fit algorithm says that scan the entire linked list and find the hole that is big enough to hold the need of the process.



The LL Representation of the above memory is.



⇒ Now say the process 'P' requires 50bytes, Now the first fit algo says that start scanning the LL and while scanning if you find a hole that is large enough to hold the process then allocate memory to the required process.

Next Fit

The Next fit will start at

⇒ Next fit is

Best Fit

⇒ Best fit Algo which can

⇒ The disadvantages

it will cause

Worst fit

⇒ This says a assign the

Quick fit

⇒ This says the and maintain

⇒ We might have

⇒ The malloc

16. GATE 2004 Q

Consider the heap regions are in

The sequence of if we use

- a) either first fit/
- b) first fit but no

(6)

Now, after allocation the LL will be

Head.

(6)

e ths.

us/Hole.

of Linked

you free
merge them
closely after
single node

find the

be First fit
scanning
be process

Next Fit

The Next Fit is same exactly as Firstfit but the difference is we will start scanning the list exactly from the point where we left earlier
 \Rightarrow Next Fit is not better than first fit

Best fit

\Rightarrow Best Fit Algo says that among all the holes find out the smallest hole which can accommodate the process.
 \Rightarrow The disadvantage of Bestfit is , it is slower compared to first fit. and it will create small small holes.

Worst fit

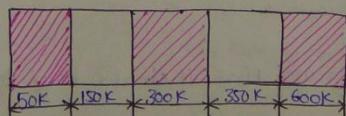
\Rightarrow This says scan the entire list and assign the biggest hole possible and assign the process.
 \hookrightarrow Not biggest enough hole

Quick fit

\Rightarrow This says that there are fixed sizes for the most frequently used process and maintain a linked list for them.
 \Rightarrow We might have to maintain separate list.
 \Rightarrow The malloc() function is going to use First fit Algorithm.

16. GATE 2004 QUESTION ON FIRST FIT AND BEST FIT

Consider the Heap in which blank regions are not in use and hatched regions are in use(occupied)



The sequence of Requests for blocks of size 300, 25, 125, 50 can be satisfied if we use

- a) either first fit / Bestfit policy (anyone)
- b) first fit but not bestfit policy
- c) Bestfit but not Firstfit policy
- (d) None



IB-GATE 98

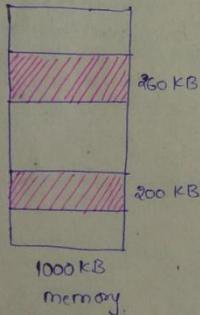
Una comput

jobs to me

17. VARIABLE PARTITIONING

A 1000KB memory is managed using variable partitions but no compaction. It currently has 2 partitions of sizes 200KB and 260KB respectively. The smallest allocation request in KB that could be denied is for

- a) 151 b) 181 (c) 231 (d) 541



Now, Total = 1000 KB

$$\text{Already occupied} = 200 + 260 = 460 \text{ KB.}$$

Now, Remaining Space = 540KB. ∴ The ~~max~~ possible ans is 541 but it is not minimum.

⇒ The max no. of partitions present in the partition is 3.

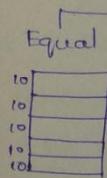
⇒ Now we need to take care that the three partitions should be minimum (this occurs when

the three partitions are of equal size)

so the remaining space = 540KB should be divided into 3 partitions of equal size = $\frac{540}{3} = 180\text{ KB}$.

∴ The min size of the programs that it should have to get rejected is

181



1F ↑

EF ↑

B-GATE 98 QUESTION ON FIXED PARTITIONING

(63)

In a computer system where the best fit algo is used for allocating jobs to memory partitions the following situation was encountered

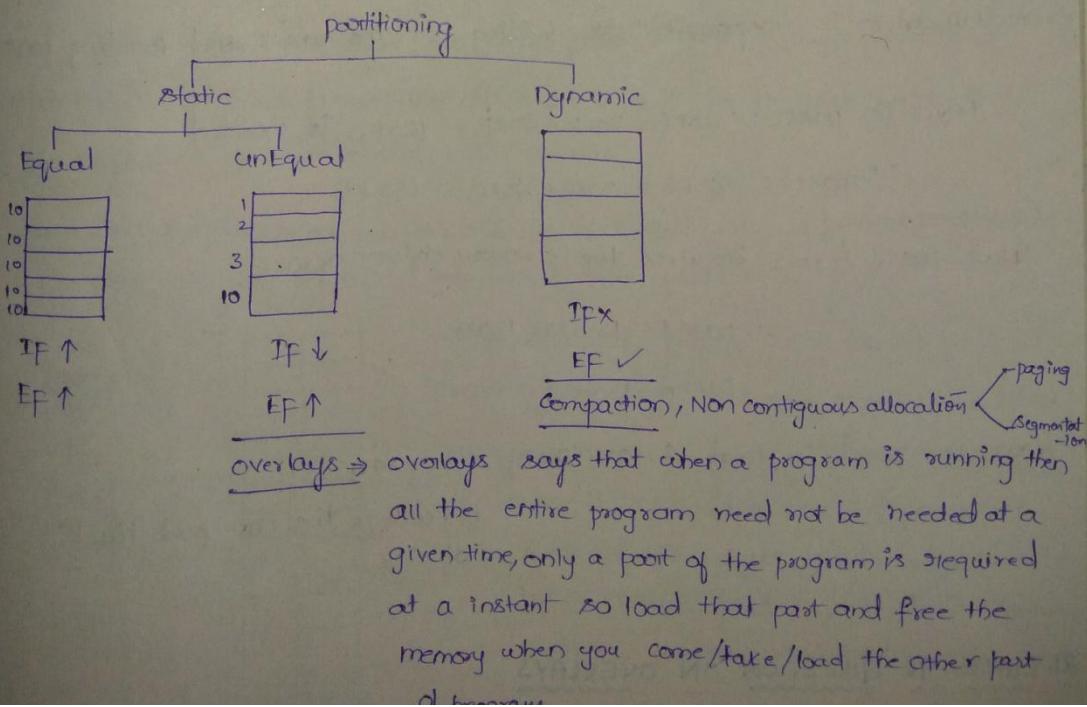
Partition sizes in KB	4K	8K	20K	2K
Jobs	2K	14K	3K	6K
Time for execution	4	10	2	1

when will 20K Job
complete?

4K	3K (0-2)	
8K	6K (0-1)	
20K	14K (0-10)	10K (10-11) At t=11, 20K (11+8=19)
2K	2K (0-4)	
partitions	Jobs	All time t=10, both 8K and 20K partition gets freed.

∴ Ans = 19

19. SUMMARY ON PARTITIONING



20. OVERLAYS

⇒ Sometimes the size of the biggest partition will be less than the size of the process. In that case we should go with "OVERLAYS"

⇒ Only a part of the program is loaded in the memory at any given instant of time.

⇒ The example of overlays is 'Assembler'

Consider a 2 pass assembler

pass1: 70KB	pass2: 80KB	Symbol table: 30KB
Common Routine: 20KB	Total memory: 200KB	

2 pass means at any time it will be doing only one thing either 1st pass/ 2nd pass

In the first pass it needs pass1 needs Symbol Table: 30 KB, Common Routine: 20 KB
 Pass1 need: $70 + 30 + 20 = 120$ KB.

⇒ But at any time only one — pass will be in use and both the passes always need symbol table and common routine. If overlay driver is 10KB, then what is the min partition size required?

⇒ Overlay driver is responsible for pulling out the pass1 and loading pass2.

Now, In pass1: $70\text{KB} + 30\text{KB} + 20\text{KB} = 120\text{KB}$ is needed

In pass2: $80\text{KB} + 30\text{KB} + 20\text{KB} = 130\text{KB}$.

Now, pass1 / pass2 requires the overlay driver of 10KB

$$\therefore \text{pass1} = 120 + 10 = 130\text{KB}$$

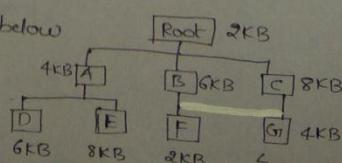
$$\text{pass2} = 130 + 10 = 140\text{KB}$$

Now, we need in total a partition of size ↓

big enough that can hold the partition of size 140.

21. GATE 98 QUESTION ON OVERLAYS

The overlay tree for a program is shown below



process

(64)

what will be the size of partition (in physical memory) required to load (and run) this program?

(65)

- a) 12KB b) 14KB c) 10KB d) 8KB.

start of

We might need $(R+A+D)$ in the memory = $2+4+6 = 12\text{ KB}$

Similarly. $(R+A+E) = 14\text{ KB}$

$(R+B+F) = 10\text{ KB}$

$(R+C+G) = 14\text{ KB}$

30KB

∴ The size of partition = $\max \{12, 14, 10, 14\}$

Min Size of partition = 14

2nd pass

routine: 20KB

of 20

120KB.

sizes

is

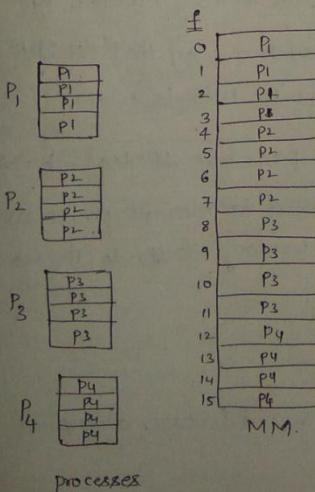
22. NEED FOR PAGING

⇒ The main problem that even the dynamic partitioning could not eliminate is External Fragmentation.

⇒ so what they have proposed is divide the process into small parts called pages and also divide the memory into small parts called frames and load each page in a single frame. (The division should be in such a way that the page size and frame size are equal).

⇒ Now, let us say the Main memory size is 16KB and we have 4 processes each of size 4KB and the MM is divided into frames each of size 1KB.

partition



⇒ Now say P₂ and P₄ went for I/O (they are removed from MM and stored on hard disk)

⇒ Now a New process P₅ of size 8KB has come. It is divided into 8 pages and these are stored at 8 vacant places created by P₂ and P₄ and thus External Fragmentation can be eliminated.

⇒ CPU generates the logical address and it is divided into 2 parts <page num, page offset>

⇒ The above logical address should be converted into physical address by Memory Manager interface.

23. PAGING EXPLAINED WITH EXAMPLE

Now Let us say the \Rightarrow MainMemory size = 64B then $2^6 = 64 \Rightarrow 6$ bits are needed in the physical address that

$$\Rightarrow \text{frame size} = 4B$$

\Rightarrow No. of frames = $\frac{64B}{4B} = 16$ frames (4 bits are enough to uniquely identify a frame in MM)

$$\Rightarrow \text{process size} = 16B, \text{ page size} = 4B$$

\Rightarrow No. of pages = $\frac{16B}{4B} = 4$ pages (2 bits are enough to identify a page)

\Rightarrow Now, the structure of the process will be

P ₀	0 1 2 3
1	4 5 6 7
2	8 9 10 11
3	12 13 14 15
4	16 17 18 19
5	20 21 22 23
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

No. of frames = 16
Process size = 0-15
 $= 16B$

\Rightarrow The P
 \Rightarrow The L

\Rightarrow Now, the structure of Mainmemory will be

and say the process is loaded into MM starting from frame 2 (say frame 0, frame 1 are occupied).

0 1 2 3
1 4 5 6 7
2 8 9 10 11
3 12 13 14 15
4 16 17 18 19
5 20 21 22 23
6
7
8
9
10
11
12
13
14
15

P₀
P₁
P₂
P₃
P₄
P₅
P₆
P₇
P₈
P₉
P₁₀
P₁₁
P₁₂
P₁₃
P₁₄
P₁₅

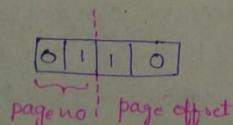
} loaded in Main Memory

MM = 64B \Rightarrow divided in frames of 8B each.

\Rightarrow Now CPU generates the logical address say 6 which means the CPU needs 6th Byte in the program but it is loaded in the memory, The 6th byte in the process should be converted to 14 in such a way that 14 identifies the 14th byte in the MM (6th byte corresponds to 14th byte in MM).

$$6 = \boxed{0 \ 1 \ 1 \ 0}$$

Now, Each page has 4 bytes init. \Rightarrow 2 bits are sufficient to identify particular byte in the memory so divide it into 2 parts



↓

Now this Address should be converted in such away that it identifies 14th Byte in MM.

\therefore with the help of above info <page no, page offset> we go to page table.

(CP)

L

Page
= 1

(66)

ie needed
physical address

no go to
identify a
MM)

ugh to.
tify a page)

frames = 4
 \Rightarrow size = $0-15$
 $= 16 \text{ B.}$

in Memory

of 8204B.

pu needs
byte in
identifies

it. \Rightarrow 2 bits
several byte
into 2

that

Every process has its own page table, the pagetable contains the frame no's
that corresponds to diff page nos.

(67)

Page No	Frame Number
0	f2
1	f3
2	f4
3	f5

<Page No> <Frame Number>

page no = $(0)_2 = (1) \Rightarrow$ Page 1

corresponds to frame 3.

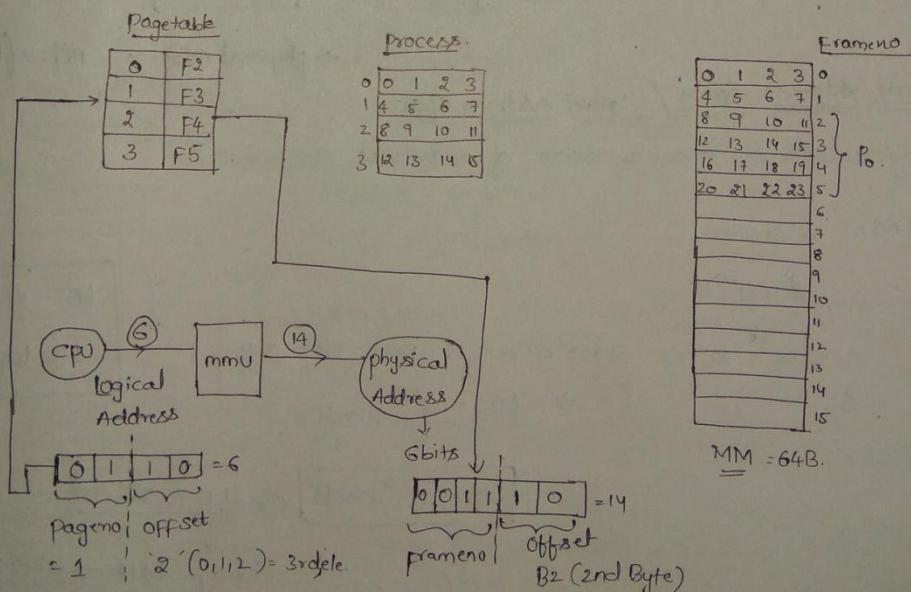
page offset = Frame offset = $(10)_2$
 $= 2$

\therefore Go to frame 3 and fetch the
Byte at 2nd position (0th, 1st, 2nd)
 $= \langle f_3 / 10 \rangle$

\Rightarrow The physical address is limited by size of MM (6 bits in this case) = 14 Bytes.

\Rightarrow The logical address size is limited by size of process (4 bits in this case)

physical address: $\boxed{0 \ 0 \ 1 \ 1 \ 1 \ 0} = 14$
 frame number offset (same as offset in logical address)
 ↓ ↓
 0011 (2)
 = 3.



24. BASICS OF BINARY ADDRESS

$$\begin{array}{lll}
 2^1 = 2 & 2^5 = 32 & 2^9 = 512 \\
 2^2 = 4 & 2^6 = 64 & 2^{10} = 1024 \\
 2^3 = 8 & 2^7 = 128 & \\
 2^4 = 16 & 2^8 = 256 & \\
 \end{array}
 \quad
 \begin{array}{lll}
 2^{10} = K & \\
 2^{20} = M & \\
 2^{30} = G & \\
 2^{40} = T & \\
 \end{array}
 \quad
 \begin{array}{l}
 \Rightarrow 128\text{GB} = 2^7 \times 2^{30}\text{B} \\
 = 2^{37}\text{B.} \\
 \Rightarrow 2^{28}\text{B} = 2^8 \times 2^{20}\text{B} \\
 = 256\text{MB.}
 \end{array}$$

Q8

page 51

1.2
Page

25. PHYSICAL ADDRESS SPACE AND LOGICAL ADDRESS SPACE

⇒ The smallest addressable unit in the Computer is word

Physical Address Space

⇒ physical Address Space is nothing but size of main memory

⇒ Now, PAS = 128KB

$$\text{PAS} = 2^7 \times 2^{10}\text{B}$$

$$\text{PAS} = 2^{17}\text{ Bytes} \Rightarrow \text{Now given } 1\text{ word} = 4\text{B.} = 2^2\text{B.}$$

$$\Rightarrow \text{PAS} = 2^{17}\text{ Bytes}$$

$$= \frac{2^{17}}{2^2} \text{ words} \Rightarrow \boxed{\text{PAS} = 2^{15} \text{W}}$$

$$\begin{aligned}
 \text{PAS} &= m \text{words} \\
 \text{PA} &= \log_2(m) \text{ bits}
 \end{aligned}$$

$$\Rightarrow \text{physical address} = 15 \text{ bits} (\log_2(2^{15}))$$

Logical Address Space / Virtual Address Space

⇒ logical Address Space = size of a process.

$$\text{LAS} = 256\text{MB}$$

$$= 2^8 \times 2^{20}\text{B}$$

$$= 2^{28}\text{B} \Rightarrow \text{Now given } 4\text{ word} = 4\text{B} = 2^2\text{B}$$

$$\Rightarrow \text{LAS} = \frac{2^{28}}{2^2} \text{ words}$$

$$\boxed{\text{LAS} = 2^{26} \text{words}}$$

$$\boxed{\text{LAS} = m \text{words}}$$

$$\boxed{\text{LA} = \log_2(m) \text{ bits}}$$

$$\downarrow \text{Logical Address} = \text{No. of bits required to address}$$

⇒ Whenever the size of the process is larger than that of MM then we are putting a part of main memory and we will be able to run it.
process in

⇒ Virtual memory was introduced to increase the degree of multiprogramming.

26. PAGI

$$\text{PAS} = M$$

$$\text{LAS} = P$$

$$\text{page size}$$

$$\text{PA} = \boxed{16}$$

$$\text{LA} = \boxed{6}$$

$$\text{Page off}$$

$$\text{PA} = \boxed{}$$

$$\text{frame} \boxed{?}$$

$$\boxed{}$$

$$\boxed{}$$

$$\boxed{}$$

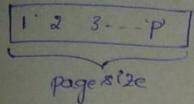
$$\text{LA} = \boxed{}$$

$$\downarrow \text{A - }$$

$$\text{Page no.} \boxed{?}$$

$$\boxed{}$$

(68) page size = framesize = 'P' words $\Rightarrow (\log_2 P)$ bits are needed



$\Rightarrow P$ words are present in the page

\Rightarrow page size = 4KB.

word size = 4B.

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

[Page size = 2^{10} words] \Rightarrow 10 bits = page offset.

(69)

26. PAGE TABLE

PAS = Main memory = M words

LAS = 128MB $\Rightarrow 2^{27}$ B. = 27 bits

LAS = process size = L words.

PAS = 1MB = 2^{20} B. = 20 bits

Page size = 'P' words.

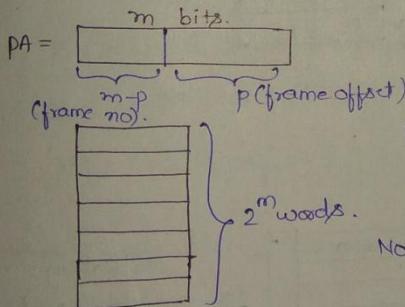
PS = 4KB = 2^{12} B. = 12 bits.

PA = $\lceil \log_2 M \rceil$ bits. = m bits (say)

LA = $\lceil \log_2 L \rceil$ bits = l bits (say)

Page offset = $\lceil \log_2 P \rceil$ = p bits (say).

$(\log_2 (2^{15}))$



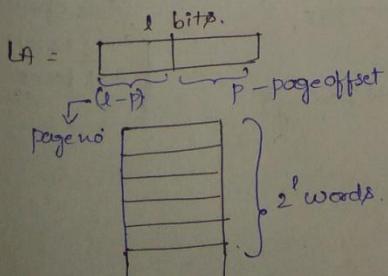
Frame = 2^p words

PAS = 2^m words

No. of frames = $\frac{PAS}{framesize} = \frac{2^m}{2^p} = 2^{m-p}$ frames

n words
 $\log_2 (n)$ bits

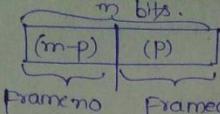
o. of bits
red to address
padding

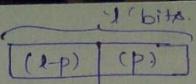


page size = 2^p words.

LAS = 2^l words

No. of pages = $\frac{LAS}{page size} = \frac{2^l}{2^p} = 2^{l-p}$ pages

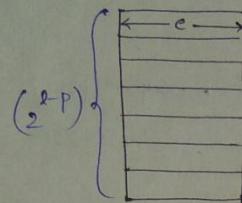
Now, PA = 

LA = 

30

27. NUI

Page table

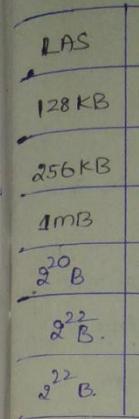


\Rightarrow No. of entries in the page table = No. of pages in the process = 2^{l-p}

\Rightarrow Let 'e' be the size of each entry

$$\Rightarrow \text{page table size} = 2^{l-p} * (e) \text{ bytes.}$$

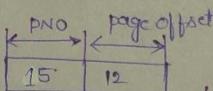
\Rightarrow If the value of 'e' is not given then, $\boxed{\text{page table size} = 2^{l-p} * (m-p)}$

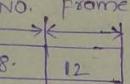


Example

$$\begin{aligned} LAS &= 128\text{MB} = 2^{27}\text{B} & LA &= 27 \text{ bits} \\ PAS &= 1\text{MB} = 2^{20}\text{B} & PA &= 20 \text{ bits} \\ PS &= 4\text{KB} = 2^{12}\text{B} & \text{page off} &= 12 \text{ bits.} \end{aligned}$$

Page table size $e = ?$

Given, LA = 27 bits \rightarrow 

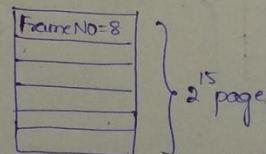
PA = 20 bits \rightarrow 

P.OFFSET = 12 bits

$$\therefore \text{No. of pages} = 2^{15}$$

$$\therefore \text{page table size} = 2^{15} \times 8 \text{ bits} = 2^{15} \times 1 \text{ byte}$$

$$\boxed{\text{PTS} \Rightarrow 2^{15} \text{ bytes} = 32 \text{ KB}}$$



Nothing is mentioned about the size of each entry so take it as frame size 2^{12} .

PTS

27. LAS

Page

$$\boxed{\begin{array}{l} \text{PTS} = 32 \text{ KB.} \\ \text{PS} = 4 \text{ KB.} \end{array}}$$

PTS > PS (Multilevel paging).

7. NUMERICAL ON PAGING

LAS	PAS	LA	PA	Pagesize	Page offset	pages	frames	PTE	PTS.
128KB	128KB	17bits	17bits	4KB	12bits	32	32	5bits	$(\frac{2^5 \times 5}{8})B$
256KB	1MB	18bits	20bits	4KB	12bits	64	2^8	2B	$(2^6 \times 2)B$
1MB	$2^{18}B$	20bits	18bits	$2^{10}B$	10bits	2^{10}	256	8bits	2^{13} bits
$2^{20}B$	512MB	20bits	29bits	$2^{12}B$	12bits	256	2^{17}	17bits	$(256 \times 16)B$
$2^{22}B$	$2^{21}B$	28bits	21bits	$2^{12}B$	12bits	2^{10}	2 ⁹	9bits	
$2^{22}B$	$2^{22}B$	22bits	22bits	$2^{14}B$	14bits	2^8	2^8	8bits	256B

$\Rightarrow LAS = 128KB \Rightarrow LA = \lceil \log_2 128 \rceil = 7 \text{ bits} = 128KB = 2^7 \times 2^{10}B = 2^{17} \text{ Bytes} = 17 \text{ bits}$

$PAS = 128KB \Rightarrow PA = \lceil \log_2 128 \rceil = 7 \text{ bits}$

page size = 4KB \Rightarrow page offset = $4 \times 2^{10}B = 2^{12}B \Rightarrow$ page offset = 12 bits.

No of pages = $\frac{LAS}{PS} = \frac{128KB}{4KB} = 32$ pages.

No. frames = $\frac{PAS}{PS} = \frac{PAS}{PS} = \frac{128KB}{4KB} = 32$ frames.

PTE = PTE contains frame nos, here frames = 32 \Rightarrow 5 bits will be needed to represent each frame uniquely.

PTS = (No. of pages) \times (PTE size) bits = $2^5 \times 5 \text{ bits} = \frac{2^5 \times 5}{8} \text{ Bytes} \cong 20B$

thing is mentioned out the size of each page so take 8 bits for frame

$\Rightarrow LAS = 1MB = 1 \times 2^{20}B = 2^{20}B \Rightarrow LA = \lceil \log_2 2^{20} \rceil = 20 \text{ bits}$

page offset = 10 bits \Rightarrow Now the page size = 2^{10} .

$\Rightarrow LAS = 1MB$, page size = 2^{10} \Rightarrow No. of pages = $\frac{LAS}{2^{10}} = \frac{2^{20}}{2^{10}} = 2^{10} =$

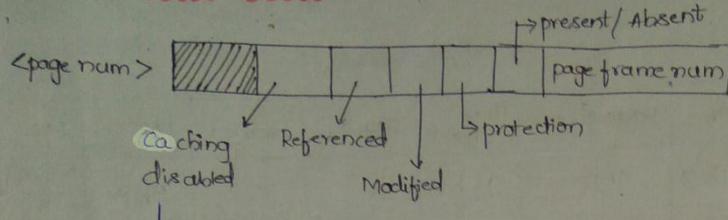
\Rightarrow Now, No. of frames = $\frac{PAS}{FrameSize} = \frac{PAS}{Page Size} \Rightarrow PAS = Page Size \times$
No. of frames

$$\begin{aligned} \Rightarrow PAS &= 2^{10} \times 256 \\ &= 2^{10} \times 2^8 = 2^{18} \end{aligned}$$

$\Rightarrow PTE = 8 \text{ bits} \Rightarrow PTS = \text{No. of pages} \times PTE$
 $= 2^{13} \text{ bits}$

$\Rightarrow PA = 18 \text{ bits}$

28. PAGE TABLE ENTRY



- ⇒ This bit/field Represent whether a particular page should be present in Cache or not.
- ⇒ Referenced: This bit say whether this page has been referenced or not.
- ⇒ Modified: whether the page has been modified or not. (Also called Dirty bit)
- ⇒ protection: what kind of protection you want to assign to a particular page (Read, write, Read/Write, ...)
- ⇒ present/Absent: Represent whether the page is present /Absent in memory.
 - ⇒ Demand paging is loading the pages into MM only when they are needed.
 - ⇒ Useful in Virtual memory concept.
- ⇒ page frame num ⇒ Contains the frame no. corresponding to particular page number.

29. GATE 2004 QUESTION ON PAGETABLE ENTRY

In Virtual memory System , size of virtual address is 32bit, size of physical Address is 30bits, page size is 4KB and size of each page table entry = 32bit, the MM is Byte addressable . Which of the following is main max no. of bits that can be used for storing protection and other information in each page table entry.

- A) 2 B) 10 C) 12 D) 14

⇒ logical Address = LA = 32 bits

$$PA = 30 \text{ bits}$$

$$PS = 4 \text{ KB}$$

$$PTE = 32 \text{ bits}$$

$$PA = 30 \text{ bit}$$

$$LA = 32 \text{ bit}$$

$$NOA, PA$$

30. NEED

$$LA = 22 \text{ bits}$$

$$LAS = 2^2 \text{ By}$$

$$\begin{aligned} PS &= 4 \text{ KB} = \\ &= 2^{12} \text{ B} \end{aligned}$$

$$PTE = 4 \text{ B}$$

$$PIS = (\text{No. of})$$

⇒ Now, To

should be

big eno

should

(CPU)

B

The

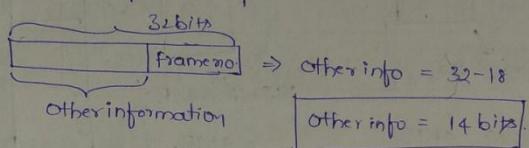
of

(72) $PA = 30 \text{ bits} \Rightarrow PAS = 2^{30} \text{ B.}$ {Size of process > Size of Main Memory}
 $LA = 32 \text{ bits} \Rightarrow LAS = 2^{32} \text{ B.}$

NOW, page size = Frame size = $4 \text{ KB} = 2^{12} \text{ B.}$

NO. of frames = $\frac{PAS}{FS} = \frac{2^{30} \text{ B}}{2^{12} \text{ B}} = 2^{18} \text{ B.}$

\Rightarrow NO. of bits required for frame no = $\lceil \log_2 2^{18} \rceil = 18 \text{ bits.}$



Now,

in memory.

M only when

30. NEED FOR MULTILEVEL PAGING

LA = 32 bits.

LAS = 2^{22} Bytes.

$PS = 4 \text{ KB} \Rightarrow \text{page offset} = 12 \text{ bits} \Rightarrow \text{No. of pages} = \frac{LAS}{PS} = \frac{2^{22} \text{ B}}{2^{12} \text{ B}} = 2^10$

particular

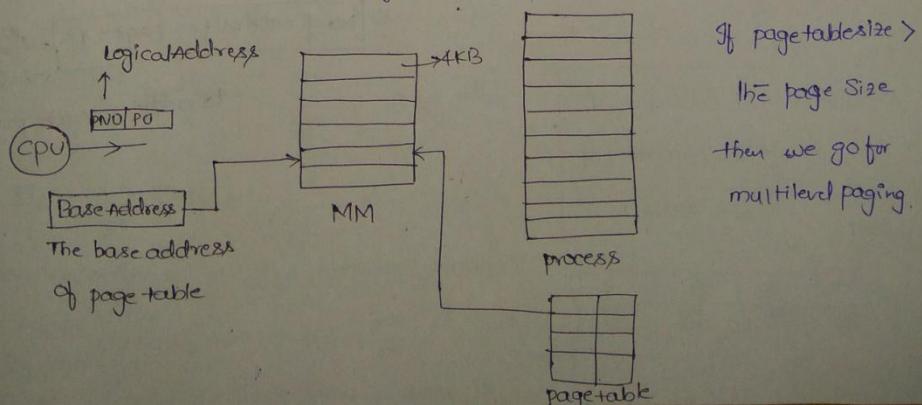
No. of pages = 1K

PTE = 4B

PT = (No. of pages) * PTE = $1K * 4B = 4 \text{ KB.}$

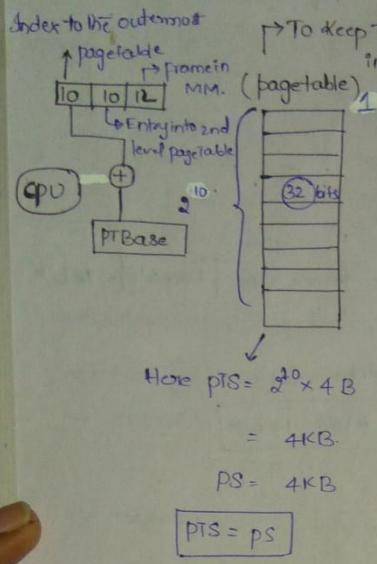
of physical
32bit, the
of bits that
page table

\Rightarrow Now, To run a process/program, the process pages and the page table should be loaded in the Mainmemory. Now if the page table is very much big enough to fit in one frame of the main memory then the page table should also be divided into pages.



31. TWO-LEVEL PAGING EXAMPLE

(14)



⇒ No. of entries = 2^{10}

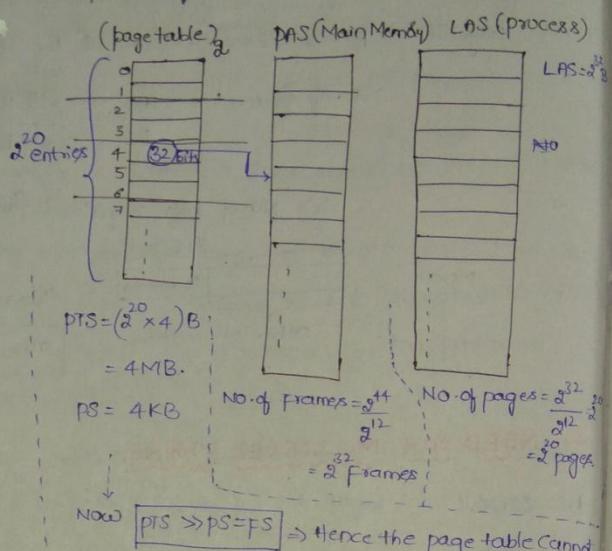
→ To identify a particular entry we need 10 bits.

⇒ This pagetable should always be in main memory.

⇒ The pt Base Register stores the Address of the frame that contains the (pagetable).

→ To keep track of pages of $PS = 4KB$ are loaded in which frames of MM.

$$PS = 4KB = FS$$



Now $PTS \gg PS = FS \Rightarrow$ Hence the page table cannot

fit in one frame of the MM

so divide the page table into no of pages

Now we get no. of pages = $\frac{4MB}{4KB} = 1K$ pages

Now the pagetable is divided into pages and loaded in main memory. Now we need to keep track of pages of the pagetable are loaded which in which frame of MM, so we need another page table.

Now here assume each entry size = 4B.

Each page size = 4KB

⇒ No. of entries per page = $1K = 2^{10}$ = 10 bits are needed

to address each page of pagetable

No. of P

$PT_2 =$

$PT_3 =$

⇒ Index the

⇒ PTE = 4B.

PS = 1MB

3> PTE = 4B, 1

32. EX

VA = LA	PO
48b	
64b	
72b	
72b	2
72b	16

⇒ VA = 4

PS = 16

$\therefore PT_1 =$

⇒ VA = 6

PS = 1N

PTE =

No. of P

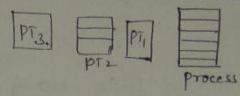
32. EXAMPLES ON MULTILEVEL PAGING

VA=LA (processes)	Page size	PTE	Address splitting		
			PT1	PT2	PT3.
48b	16KB	4B	$(2^{34} \times 4)B$	$2^{22} \times 2^2 B = 2^{24} B$	$2^4 B$
64b	1MB	4B	2^{46}	$2^{28} B$	$2^{10} B$
72b	1GB	4B	$(2^{42} \times 2^2)B$	$(2^{14} \times 2^2)$	-
72b	256MB	4B	$(2^{44} \times 2^2)B$	$(2^{18} \times 2^2)$	-
72b	16MB	4B	$(2^{48} \times 2^2)B$	$(2^{26} \times 2^2)$	$(2^4 \times 2^2)$

$\Rightarrow VA = 48\text{ bits} \Rightarrow VAS = 2^{48} B$

$\text{Pages} = \frac{2^{32}}{2^{12}} = 2^20$
= 2^{20} pages

$PS = 16\text{ KB} = 2^{14} B \Rightarrow \text{No. of pages} = \frac{VAS}{PS} = \frac{2^{48}}{2^{14}} = 2^{34} \text{ pages. } \times (PTE, size) = 2^{34} \times 2^2$
 $\therefore PT_1 = 2^{36} B, \text{ Now, } PS = 16\text{ KB} = 2^{14} B$



$\Rightarrow PT_2 = \frac{PT_1}{PS} = \frac{2^{36}}{2^{14} \times 2^{10}} = 2^{22} B \times PTE = 2^{22} \times 2^2 B = 2^{24}$

No. of pages in PT_2

$PT_1 = (\text{No. of pages}) \times (2^{12} \text{ of each entry in } PT_1)$

$\Rightarrow PT_2 = \left(\frac{PT_1}{PS} \times PTE \right) = \frac{2^{36}}{16 \times 2^{10}} \times 4B = 2^{12} B$

$PT_2 = \left(\frac{PT_1}{PS} \right) \times PTE$

Gives no. of pages in PT_2

$\Rightarrow VA = 64\text{ bits} \Rightarrow VAS = 2^{64} B$

$PS = 1MB \Rightarrow 2^{20} B$

$PTE = 4B$

No. of pages = $\frac{VAS}{PS} = \frac{2^{64}}{2^{20}} = 2^{44}$ $\Rightarrow PT_1 = \frac{\text{No. of pages} \times PTE}{2^{44} \times 4B} = 2^{46} B$

$2^{10} = 10 \text{ bits}$

needed page of pagetable

$PT_2 = \frac{PT_1}{PS} \times PTE = \frac{2^{46}}{2^{20}} \times 2^2 B = 2^{28} B$

$PT_3 \text{ entries} = 2^8 \text{ (No. of pages)}$

$PT_3 = \frac{PT_2}{PS} \times PTE = \frac{2^{28}}{2^{20}} \times 2^2 B = 2^{10} B$

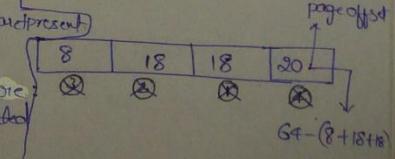
$PT_2 \text{ entries} = 2^{26}$

$PT_1 \text{ entries} = 2^{44}$

\Rightarrow Index the outermost $PT = PT_3 = 8 \text{ bits (2}^8 \text{ pages and present)}$

$\Rightarrow PTE = 4B, PS = 1MB \Rightarrow \text{No. of entries per page} = \frac{2^{20}}{2^4} = 2^{16} = 18 \text{ bits are needed}$

$\Rightarrow PTE = 4B, PS = 1MB \Rightarrow \text{No. of entries per page} = 18$.



34. Examples on how to make page table fit in one page

(76)

(2) Which

$$1) VAS = 4MB$$

$$PS = 4KB.$$

$$PTE = 4B.$$

$$VAS = 4MB = 2^{22} B.$$

$$PS = 4KB = 2^{12} B.$$

$$PTE = 4B.$$

$$\left. \begin{array}{l} \text{No. of pages} = \frac{VAS}{PS} = \frac{4MB}{4KB} = 2^{10} B \\ = 1K \text{ pages} \end{array} \right\}$$

$$\text{Now, PTS} = \text{page table pages} * e$$

$$PTS = 1K * e$$

- a) Faster
 - b) process
 - c) linked
- in the program

Now, $PTS \leq PS$ (to fit PTS in one page)

$$1K * e \leq PS$$

$$e \leq \frac{2^{12}}{2^{10}}$$

$$e \leq 4B.$$

37. GATE

$$PA = 36 \text{ bits}$$

as follows

→ Bits : 3

→ Bits : 8

→ Bits : 12

→ " "

Then PTE

A) 20, 20, 20

$$\text{Now, } PTS = \text{No. of pages} * e$$

$$PTS = 32K * e$$

$$\Rightarrow PTS \leq PS$$

$$\Rightarrow 2^{15} * e \leq 2^{17}$$

$$e \leq 4B$$

35. GATE Q1 AND Q9 QUESTION ON PAGING

1) Consider a machine with 64MB physical memory and 32-bit virtual address space. If the page size is 4KB, what is approximate size of PT?

- a) 16MB b) 8MB c) 2MB d) 24MB

$$PAS = 64MB = 2^{26} B$$

$$VA = 32 \text{ bits} \Rightarrow VAS = 2^{32} B$$

$$PS = FS = 4KB = 2^{12} B$$

$$\left. \begin{array}{l} \text{No. of pages} = \frac{2^{32}}{2^{12}} = 2^{20} \text{ pages} \end{array} \right\}$$

$$e = \text{frame no.} = (26 - 12)$$

$$\Rightarrow PTS = \text{No. of pages} * e$$

$$= \text{No. of pages} * \text{frame size num}$$

$$= 2^{20} * \text{frame num}$$

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

$$= \frac{14M}{8} \text{ Bytes} = 2MB$$

⇒ when we
in all the

38. GATE 20

Theory ques

41. GATE 20

$$VA = 46 \text{ bits}$$

$$PTE = 32 \text{ bits}$$

Let size of pa

(76)

Q. Which of the following is/are advantages of VM?

(77)

- $= 2^{10} \text{ pages}$
- a) Faster access to memory on an average (All pages are in hard disk so slower)
 - b) process can be given protected address spaces (Not specific to VM)
 - c) linker can assign address independent of where the program will be loaded in the physical memory. (Relocation, this exists in contiguous Allocation also)
 - d) programs larger than the physical memory size can be run.

ns in one page

3. GATE 2008 QUESTION ON MULTILEVEL PAGING

PA = 36 bits, VA = 32 bits, PS = 4KB, PTE = 4B, 3-level PT is used. VA is divided as follows

- Bits 30-31 are used to index I level PT
- Bits 21-29 " " " " II " "
- Bits 12-20 " " " " III " "
- " 0-11 are used as offsets within a page

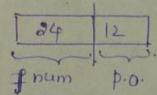
Then PTE sizes in I, II and III level PT are:

- A) 20, 20, 20 B) 24, 24, 24 C) 24, 24, 20 D) 25, 25, 24

PA = 36 bits

at Address

PS = 4KB \Rightarrow page offset = 12 bits



⇒ when we have multi-level paging we use same no. of bits to represent a frame in all the levels.

3. GATE 2009 QUESTION ON MULTILEVEL PAGING

Theory question easily answerable

frame num

4. GATE 2013 QUESTION ON MULTILEVEL PAGING

M bits

VA = 46 bits, PA = 32 bits, 3-level paging, first level table has exactly one page

PTE = 32 bits, what is the size of the page in computer?

Let size of page = P bytes, and PTE = 4B. \Rightarrow No. of entries per page = $P/4$

$$\begin{aligned} \text{Page} & \left\{ \begin{array}{l} \text{P/4 entries come out} \\ \text{P/4 entries will come out} \end{array} \right. \Rightarrow \text{No. of entries} = (P/4)^3 \quad \therefore VAS = (P/4)^3 * P = 2^{46} \\ & \qquad \qquad \qquad \qquad \qquad \qquad \qquad \boxed{P = 8 \text{ KB}} \end{aligned}$$

42- FINDING OPTIMAL PAGE SIZE

$$VAS = 4GB$$

$$PS = 4KB$$

$$\Rightarrow \text{No. of pages} = \frac{VAS}{PS}$$

$$= 1M \text{ pages}$$

$$VAS = 4GB$$

$$PS = 4MB$$

$$\text{No. of pages} = \frac{VAS}{PS}$$

$$= \frac{4GB}{4MB} = 1K \text{ pages}$$

Therefore $PS \uparrow$ $PTS \downarrow$

$$\text{Page size} = p \text{ Bytes}$$

$$PTE = e \text{ bytes}$$

$$\text{on Average } VAS = S \text{ Bytes}$$

$$\text{overheads} = (P_2) + (S_p) * e$$

The last page might not be completely filled

$$\text{To minimize the overheads, differentiate w.r.t. } p \Rightarrow \frac{d}{dp} (P_2 + (S_p) * e) = 0$$

$$\Rightarrow \frac{1}{2} - S_p * e = 0$$

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

| S: Avg. process size
e: PTE

optimal page size

S	e	Page size
4KB	8B	256B
16MB	8B	8FB
256GB	32B	4MB

$$\triangleright VAS = 4KB = 2^2 B$$

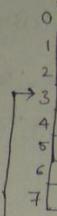
$$PTE = e = 8B$$

$$\Rightarrow p = \sqrt{2 \times 2^2 \times 8} = 2^8 = 256B$$

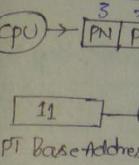
\Rightarrow In general the OS takes the process and calculate the overhead based on that formula, But some OS takes the process, divides them into sections (code section, data section, stack section) and divide each section into pages based on the above formula. Then the Overhead on a particular section will be

$$\text{Overhead} = \left[\frac{np}{2} + s(p) * e \right]$$

n = no. of pages the section is divided into.



43. VM



\Rightarrow The VM c

\Rightarrow VM is

\Rightarrow The process

MM, Sh

the page

is design

P2 and

44. TLB

Now, If VAS =

\Rightarrow N

further divided

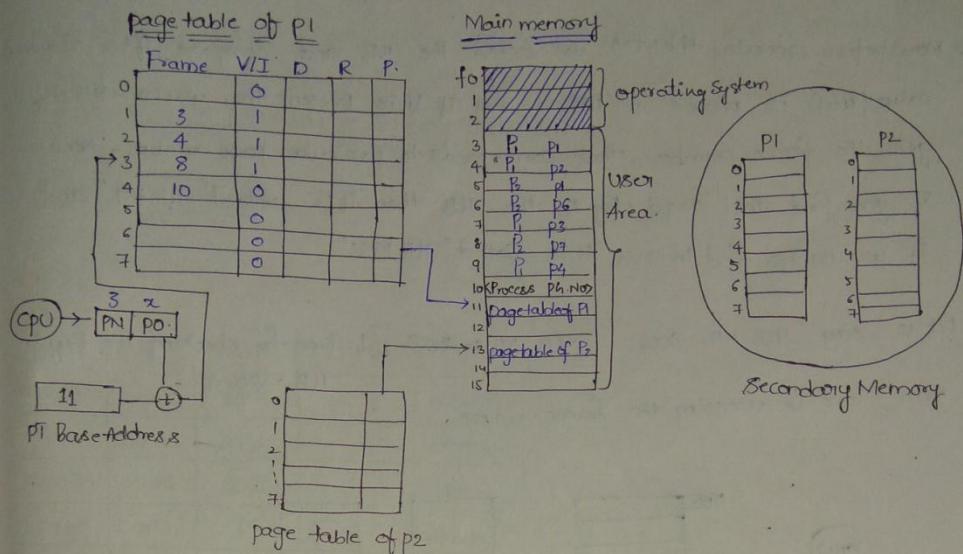
the disad van

b access a frame

going to take a lot

43. VM INTRODUCTION

TS ↓



⇒ The VM concept is used when the process size is greater than the MM.

Avg. process
size
 \approx PTE

⇒ VM is provided by operating system but overlays is by Manual intervention

⇒ The process in the Secondary memory is divided into pages and allocated in the

MM, In the MM $\langle P_i, p_i \rangle$ represent \langle process no, page no \rangle , Now to keep track of

the pages of program are loaded in which frame number is the page table

is designed, and when there is context switch the control switches from P_1 to

P_2 and the value in "PT Base Address" will now be 13.

44. TLB

Now, If $VAS = 4GB$, $ps = 4B$

$$\Rightarrow \text{No. of pages} = 1 \text{ million.}$$

} Now, the size of the page table will contain 1 million entries and it cannot be loaded in main memory. So the page table is also

further divided into pages and this concept is called multilevel paging but the disadvantage with this method is the memory access time, in order

to access a frame in MM we have to access the series of page tables and it is going to take a lot of time. So they have discovered a memory (Not Registers & MM) called (CACHE) OR(TLB) and the page table is loaded in Cache (Translation lookaside buffer)

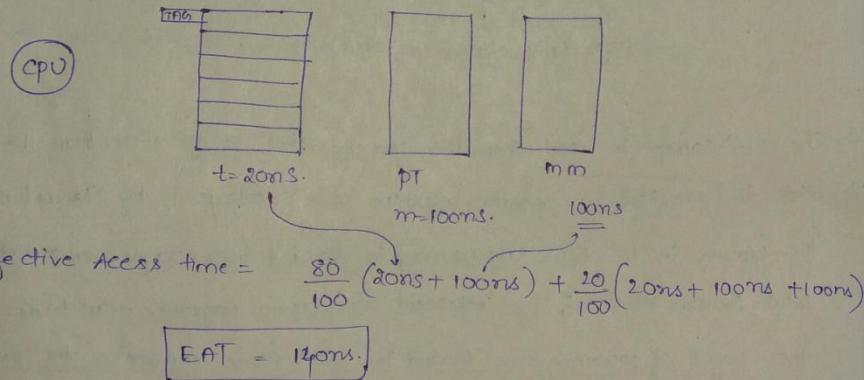
based on
sections.
into pages
last section
the section
into.

⇒ The cache is faster than MM and cheaper than Registers.

⇒ The TLB is also sometimes called Associative ↑ because, every ele of this memory is having 'TAG'.

⇒ Now before accessing the MM we access the TLB and compare if the required value (Key) is present in TLB or not if it is present then you can directly get the frame number, that corresponds to particular page number. Now if you find the reqd key in the TLB then it is called "TLB HIT" and if you cannot find in TLB it is called "TLB MISS".

Let us say TLB HIT = 80% TLB MISS = 20%, t = time for checking the key in TLB = 20ns.
m = time for accessing the frame in MM.



If TLB HIT = p ⇒ TLB miss = 1-p

$$\text{EAT} = p(t+m) + (1-p)(t+2m)$$

EAT = $p(t+m) + (1-p)(t+2m)$ (Single level page Table)

If there are K levels then $\boxed{\text{EAT} = p(t+m) + (1-p)(t+km+m)}$

45. GATE

A paging takes 50

A) 54

⇒ EAT =

EAT -

46. NUM

TLBA	n
20ns	1

!> EAT =

2> EAT =

3> Ema

→ $\frac{20}{10}$

⇒

=

(80)

45. EAT 2008 QUESTION ON TLB

(81)

of this

be required

directed
Now

" and

A paging scheme uses TLB. A TLB access takes 10ns and main memory access takes 50ns. What is the EAT (in ns) if TLB hit ratio = 90% and there is no page fault.

- A) 54 B) 60 C) 65 D) 75.

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

$$EAT = 65 \text{ ns}$$

46. NUMERICALS ON TLB

be key in

TLBA	MA	TLB H	PT levels	EMAT
20ns	100ns	80%	1	140ns
20ns	100ns	80%	2	160ns
20ns	100ns	80%	3	180ns
20ns	100ns	90%	1	130ns
20ns	100ns	60%	1	160ns
20ns	100ns	50%	1	170ns

$$1) EAT = \frac{80}{100} (20 + 100) + \frac{20}{100} (20 + 100 + 100)$$

$$= 96 + 24 = 140 \text{ ns.}$$

$$2) EAT = \frac{80}{100} (20 + 100) + \frac{20}{100} (20 + 2(100) + 100)$$

$$= \frac{80}{100} \times 120 + \frac{20}{100} \times (320) = 96 + 64 = 160 \text{ ns.}$$

$$3) EMAT = 130 \text{ ns.}$$

$$\Rightarrow \frac{20}{100} (20 + 100) + \left(100 - \frac{x}{100}\right) (20 + 100 + 100) = 130 \quad p(20+100) + (1-p)(20+100+100) = 130$$

$$= 160 = 0.6(t+100) + (0.4)$$

$$(t+100+100)$$

$$170 = 0.5(20+m) + 0.5(20+2m)$$

$$2m)$$

$$\Rightarrow \frac{x}{100} (120) + \frac{100-x}{100} (220) = 130$$

$$= \frac{6x}{5} + \frac{(104-x)}{5} = 130$$

$$x = 90 \text{ ns%}$$

$$p(20+100) + (1-p)(200) = 130$$

$$120p + 120 - 20p = 130$$

$$= 100p = 10 \Rightarrow p = 10/100 = 10\%.$$

47. TLB SUMMARY

(82)

TLB access time = t

Main memory access = m

TLB miss rate = $p \Rightarrow$ TLB Hitrate = $(1-p)$

$$EMAT = p(t+m+t) + (1-p)(t+m)$$

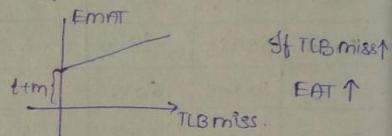
$$= pt + 2pm + t + m - pt - pm$$

$$\boxed{EMAT = t + m + p(m)} \Rightarrow \text{let } EMAT = y$$

$t+m = \text{Constant} = C$

$$p(m) = \frac{pxm}{x} \downarrow \downarrow \text{Represented by } a$$

$$\Rightarrow y = c + ax \Rightarrow y = ma + c$$



If no. of levels = K then

$$EMAT = p(t + km + m) + (1-p)(t+m)$$

$$= p(t + (1+k)m) + (1-p)(t+m)$$

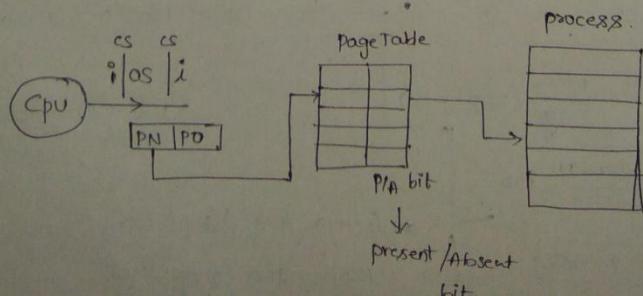
$$= pt + (1+k)pm + t + m - pt - pm$$

$$\boxed{EMAT = t + m + p(km)} \quad p = \text{miss rate not hit rate.}$$

48. PAGE FAULT

\Rightarrow Referring to a page that is not present in the memory is called PF.

\Rightarrow Demand paging = Don't load any page until it is required



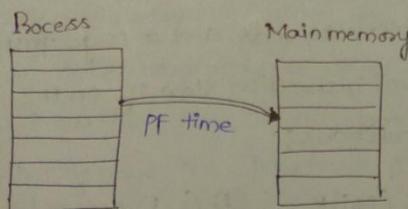
Now, the CPU generates the logical address and with that address we go to corresponding entry in the pagetable now if the entries correspondingly P/A bit is 0 then it says that the page that you are requesting is not loaded in the main memory. Then there will be context switch (CS) from user process to operating system, now the OS try to load the required page from the secondary storage and the context is again switched to the user process.

\Rightarrow No. of page references = No. of page faults then it is called Thrashing.

\Rightarrow EMAT in case of page fault is. = $p * (\text{service time} / \text{page fault time}) + (1-p)(\text{memory access time})$

$$\boxed{\text{EMAT} = p * (\text{service time}) + (1-p)(\text{memory Access time})}$$

↓
Also called page Fault time



⇒ Now, initially the process is going to start without loading any page in MM then CPU whenever it generates 'LA' it will find out that the 'pageno' is not in MM, then page should be loaded

into MM and again the same instruction will start which will generate the same logical address, the entire time (To stop the process, copy the page, again start the time is called page fault time). and again after loading again in need to refer to the page in Memory.

$$\boxed{\text{EMAT} = p * (\text{Service time} + \text{ma}) + (1-p)(\text{ma})}$$

↓
Very Negligible

49. GATE 2011 QUESTION ON PAGE FAULT

(84)

Given MA

Se

H

E

Let the pf service time be 10ms in a computer with average memory access time being 20ns. If one page fault is generated for every 10^6 memory access, what is the EAT for the memory?

- A) 21ns B) 30ns C) 23ns D) 35ns

$$EMAT = p * (\text{servicetime} + \text{MA}) + (1-p)(\text{MA})$$

Given

$$p = \frac{1}{10^6}, \text{MA} = 20\text{ns}$$

$$\text{servicetime} = 10\text{ms}$$

$$= p * (\text{servicetime}) + [p * (\text{MA}) + (\text{MA}) - p * (\text{MA})]$$

$$= p * (\text{servicetime}) + (\text{MA})$$

$$= \frac{1}{10^6} * (10\text{ms}) + ((20\text{ns})) \rightarrow \text{Access time without page faults.}$$

$$= 10\text{nsec} + (20\text{ns})$$

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

50. GATE 98 QUESTION ON PAGE FAULT

If an instruction takes 'i' micro seconds and a page fault takes an additional 'j' usec, the effective instruction time if, on the average a page fault occurs every K instructions?

- A) $i + (j/k)$ B) $i + j * k$ C) $(i+j)k$ D) $(i+j)*k$

$$EMAT = p * (\text{servicetime}) + \text{MA}$$

$$EIT = i + \frac{1}{k}(j) = \text{instruction access time} + p * (\text{servicetime})$$

$$\boxed{EIT = i + (j/k)} = i + p * (j)$$

$$= i + \frac{1}{k} * j = (i + j/k)$$

⇒ Service

"

MA =

$\boxed{EMAT = K * i}$

$\boxed{3 = }$

$\Rightarrow 1 + 100$

$= 200\mu\text{s}$

51. GATE 2000 QUESTION ON PAGE FAULT

Suppose the time to service a page fault is on average 10msec, while a memory access takes 1μsec. Then a 99.99% hit ratio results in avg. memory access time of?

- A) 1.9999 msec B) 1 msec C) 2 msec

(84)

Given $MA = 1\mu\text{sec}$

memory access

memory access,

 $mf = 20\mu\text{s}$ $me = 10\mu\text{s}$ Service time = $10\mu\text{sec}$ Hit ratio = 99.99% \Rightarrow miss rate = 0.01% $\Rightarrow 0.0001$

$$\therefore EMAT = P * (\text{Service time}) + MA = (0.0001)(10 \times 10^{-3})$$

$$= 1\mu\text{sec} + (10^{-4} \times 10^{-2})$$

$$= 1\mu\text{sec} + 1\mu\text{sec}$$

$$\boxed{EMAT = 2\mu\text{sec}}$$

(85)

Q2. GATE 2007 QUESTION ON PAGE FAULT AND DIRTY BIT

A demand paging system takes 100 time units to service a page fault and 300 time units to replace a dirty page. Memory access time is 1 time unit. The prob. of page fault is 'P'. In case of page fault the prob. of page being dirty is also 'P'. It is observed that the average access time is 3 time units. Then value of 'P' is

- a) 0.194 b) 0.233 c) 0.514 d) 0.981

⇒ Here the Dirty page represent when you are accessing a particular page. Check whether it is modified/not using dirty bit. If you find it is modified then writeback the page in the process and then replace it. So when there is a dirty page the pagefault service time will increase.

⇒ Service time/page fault time (in case of no dirty pages) = 100 time units.
 " " " " (" " " Dirty pages) = 300 time units.

$$MA = 1 \text{ time unit}$$

$$PS = (1-p)(100) + p(300)$$

$$\boxed{PFT = 100 + 200p}$$

$$EMAT = 1 \text{ unit} + m + (p)(PFT)$$

$$3 = 1 + (p)(100 + 200p)$$

$$\Rightarrow 1 + 100p + 200p^2 = 3$$

$$\Rightarrow 200p^2 + 100p - 2 = 0 \Rightarrow p = 0.0194$$

53. GATE 2003 QUESTION ON TLB AND PAGING

A processor uses 2-level paging $VA = PA = 32$ bits, Byte addressable

VA:

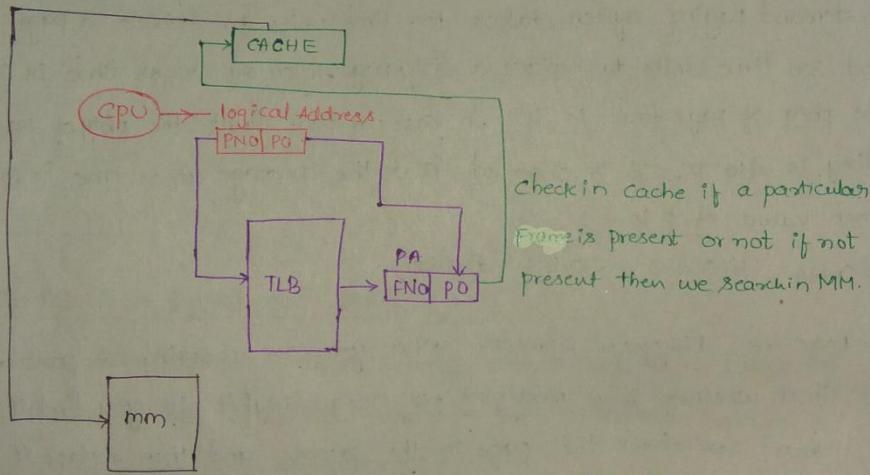
10	10	12
----	----	----

 PTE = 4 bytes, TLB has hit rate of 96%, Cache has hit

rate of 90%. Main memory access time is 10ns, Cache access time is 1ns and TLB access time is 1ns.

- a) Assuming no page faults, the average time taken to access a VA is approximately (to the nearest 0.5 ns)

- a) 1.5 ns b) 2 ns c) 3 ns d) 4 ns



$$EMAT = \text{Avg. time taken to access Virtual Address} = (VA \rightarrow PA) + (\text{fetch the word from process})$$

Take the help of TLB
First check in Cache

If not present then go to MM.

$$\Rightarrow \text{Avg. time} = t + \underbrace{(1 - p_t)(k * m)}_{\substack{\uparrow \text{No. of levels} \\ \text{conversion from } (VA \rightarrow PA)}} + \underbrace{c + (1 - p_c)(m)}_{\substack{\uparrow \text{Hit rate of Cache} \\ \text{fetching}}}$$

$$= 1 \text{ ns} + \frac{4}{100} (2 * 10) + 1 \text{ ns} + \frac{10}{100} (0)$$

Avg. time = 3.8 ns

- a) Suppose
two cont
contiguous
virtual
addressing
a) 8KB

Now, Addr:

$$\Rightarrow PTE = PTS$$

$$PTS$$

$$\Rightarrow 2 \text{ pages} =$$

54. GATE 2003

Consider a
access to
instruction
ratio is
is the effe

- a) 6.45 ns

$$EMAT = CP$$

$$EMAT = (V$$

(8)

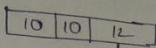
- Q) Suppose a process has only the following pages in virtual address space
 two contiguous code pages starting at virtual address 0x00000000, two
 contiguous data pages starting at VA 0x00400000 and a stack page at
 virtual address 0xFFFFFFF000. The amount of memory required for
 storing page tables of this process

(87)

- a) 8KB b) 12KB c) 16KB d) 20KB

VA is

Now, Address split

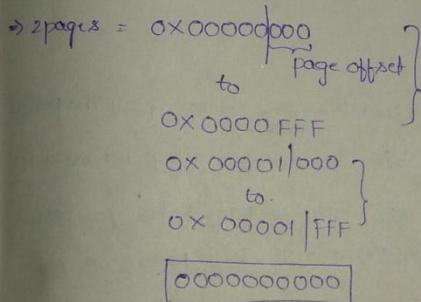


Represent page offset \Rightarrow size of page = 2^{12} B.
 $= 4KB.$

$$\Rightarrow PTE = PTS = \text{No. of pages} * \text{Size of each PTE}$$

$$PTS = 4MB.$$

particular,
 st if not
 main MM.



2 pages: 0x 004001000
 0x 004001FFF
 0x 00401000
 0x 00401FFF
 $004 = \boxed{00000000100}$

Search the word
 (process)
 Pick in Cache
 out then go

64. GATE 2004 QUESTION ON TLB AND PAGE FAULT

Consider a system with 2-level paging scheme in which a regular memory access takes 10nsec and servicing a page fault takes 8msec. An average instruction takes 100 nsec of CPU time and two memory access. The TLB hit ratio is 90% and page fault rate is one in every 10,000 instructions. What is the effective average instruction execution time?

- A) 645ns B) 1050 ns C) 1215ns D) 1230 nsec

$$EAIT = CPUTime + 2 * (EMAT)$$

$$EMAT = (VA \rightarrow PA) + (\text{Access time from PTA})$$

$$= t + (1 - p_f)(2 * MAT) + m + p_f * PS$$

$$= 0 + (0.1)(2 * 150n) + 150n + \left(\frac{1}{10,000}\right)(8m)$$

$$= (30 + 150 + 800) \mu\text{sec} = 980 \mu\text{sec} \Rightarrow EAIT = 100ns + 2 * (980) = 2060ns.$$

55. GATE 2014 QUESTION ON TLB

Consider a paging HW with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10msec to access the TLB and 80msec to search the physical memory. If the TLB hit rate ratio is $0.6 = 60\%$, the EMAT is ____?

$$\begin{aligned}\text{EMAT} &= p(t+m) + (1-p)(t+2m) \\ &= 0.6(10+80) + (0.4)(10+160) \\ &= \frac{60}{160} \times 90 + \frac{40}{160} \times 170 \\ &= 54 + 68 \\ &= 122\text{msec.}\end{aligned}$$

56. INVERTED PAGE TABLE

Now, assume there are 10 processes running now the pagetables of all the process should be loaded now, the main memory will be occupied by 10 pagetables which will take more space so the concept of Inverted page table has been proposed. The indexes in the Inverted page table will be Frame no not the page no.

Inverted pagetable

Frameno	
0	(P_1, P_2)
1	(P_1, P_3)
2	(P_1, P_5)
3	(P_2, P_4)
4	(P_2, P_3)
5	(P_2, P_4)
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

$\langle \text{process no, page no} \rangle$

page	PT of P_2
0	X
1	X
2	f_4
3	f_5
4	f_6
5	X
6	X

page	PT of P_1
0	X
1	f_1
2	X
3	f_2
4	X
5	f_3
6	X

$$\left. \begin{array}{l} \text{If } PA = 32 \\ PS = 4\text{ KB} \\ \text{IPT} = 4B. \end{array} \right\} \text{what is the size of IPT?}$$

$$\begin{aligned}\text{IPTS} &= (\text{No. of frames}) * \text{IPT} \\ &= \frac{2^{32}}{2^{12}} \text{B} * 4\text{B} \\ &= 2^8 * 4\text{B.} \\ &\Rightarrow 4\text{ MB.}\end{aligned}$$

57. IMPORTANCE

Equal Allocation
→ 30 frames
3 processes
 $\Rightarrow 10 \text{ frames} = 10 \text{ frames}$

\Rightarrow when all the it wants to used.

58. PAGE RE

Optimal page

Least Recently
time

FIFO: First

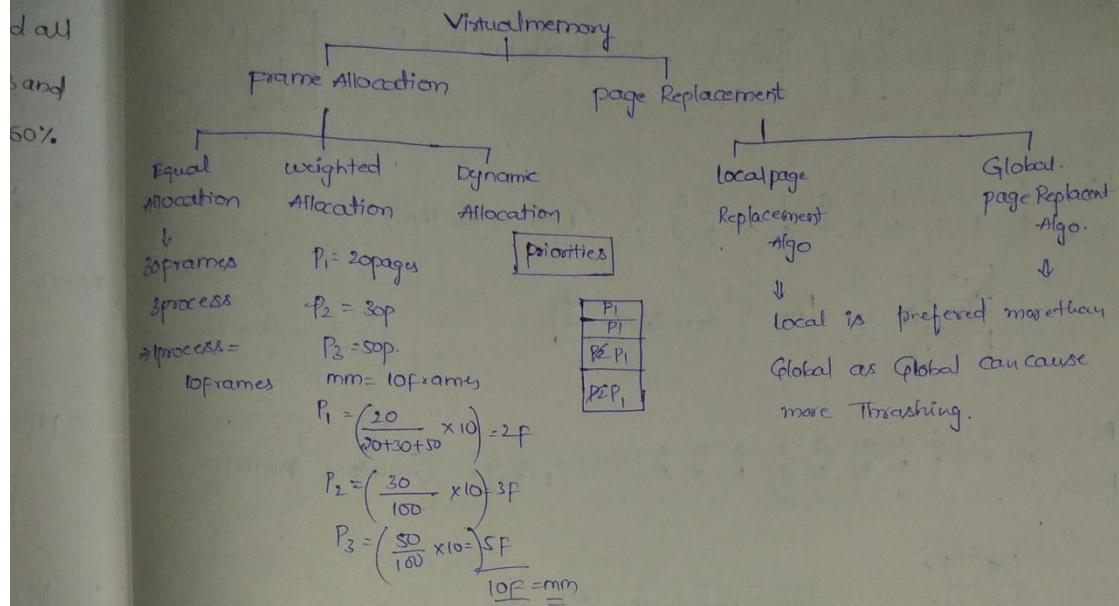
59. QUESTION

Demand Paging

GATE- 2014

Reference String =
Frames = 3.

57. IMPORTANCE OF FRAME ALLOCATION AND PAGE REPLACEMENT



→ when all the frames allocated to a particular process gets filled and if it wants to load another page of it then page replacement algorithms are used.

58. PAGE REPLACEMENT

Optimal page replacement: Replace the page that will not be referred for a long time (least no. of page faults) (Used as Benchmark)

Least Recently Used: Replace the page which has not been referred for a long time

FIFO: First Inserted page should be replaced.

59. QUESTION ON PAGE REPLACEMENT ALGORITHMS 1

Demand paging is used find the No. of page faults

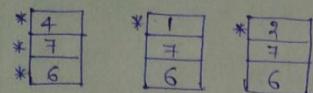
GATE-2014

Reference string = 4, 7, 6, 1, 7, 5, 1, 2, 7, 2

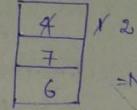
Frames = 3.

RS = 4 7 6 1 7 6 1 2 7 2 (optimal page Replacement Algo)

F=3 \Rightarrow



No. of page faults = 5



$$\begin{aligned} &= \text{No. of letters} \\ &= 4, 1, 2, 7, 6 \end{aligned}$$

$$= 5 \text{ PF}$$

G1. GAT

$$RF = 1, 2,$$

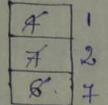
$$F = 3$$

Optimal

$$RF = 1, 2,$$

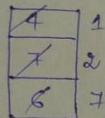
$$F = 3$$

RS = 4 7 6 1 7 6 1 2 7 2 } (LRU Algorithm)



No. of page faults = 6

RS = 4 7 6 1 7 6 1 2 7 2 } (FIFO) (maintain Queue)



No. of page fault = 6

Queue A 7 | 6 | 1 2

QUESTION ON PAGE REPLACEMENT EXAMPLE 2

(2).

0100, 0200, 0430, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0370

100 Records per page with 1 free main memory frame.

The above no. represents the decimal no. \Rightarrow now we need to convert them into page numbers.

$$0100 = 01$$

$$0499 = 04$$

$$0120 = 01$$

$$0320 = 03$$

$$0200 = 02$$

$$0510 = 05$$

$$0220 = 02$$

$$0370 = 03$$

$$0430 = 04$$

$$0530 = 05$$

$$0240 = 02$$

$$0260 = 02$$

$$0560 = 05$$

$$0200 = 02$$

$$0320 = 03$$

$$0100 = 01$$

RS = 01, 02, 04, 04, 05, 05, 05, 01, 02, 02, 02, 03, 03

P=1 $\uparrow \uparrow \uparrow$

\Rightarrow 1 0 4 5 1 7 3

\Rightarrow No. of page faults = 7 \Rightarrow No. of misses

\Rightarrow Miss Rate = $7/13 = \text{PFRATE}$

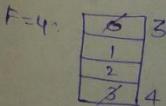
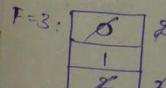
\Rightarrow No. of references made = 13. (length of RS)

64. BEUP

$$RF : 0$$

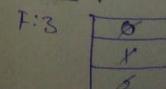
$$F : 3$$

Optimal



$$PF = 6$$

FIFO



$$PF = 6$$

90

Hitrate = 1 - missrate

$$= 1 - \frac{7}{13} = \frac{6}{13}.$$

91

No. of letters
4, 1, 2, 7, 6
= 5 PF
 $F = 3$

61. GATE QUESTION ON PAGE REPLACEMENT ALGORITHM 3

RF = 1, 2, 3, 2, 4, 1, 3, 2, 4, 1

optimal

RF = 1, 2, 3, 2, 4, 1, 3, 2, 4, 1

 $F = 3$

1	
2	4
3	2

No. of page faults = 5,

$$PFR = \frac{5}{10}$$

LRU

RF = 1, 2, 3, 2, 4, 1, 3, 2, 4, 1

 $F = 3$

X	A2
2	1
3	4

No. of page

faults = 9

$$PFR = \frac{9}{10}$$

FIFO

1	4
2	1
3	2

No. of page faults = 10

$$PFR = \text{missrate} = \frac{6}{10} = 60\%$$

1	2	3	4	1	2
---	---	---	---	---	---

$$\text{Hitrate} = 1 - \frac{6}{10} = \frac{4}{10} = 40\%$$

64. BELADY ANAMOLY

Q370

RF: 0 1 2 3 0 1 4 0 1 2 3 4

 $F = 3$.

st-them

optimal

0	2	3
1		
2	3	4

Page faults = 7

 $(6 < 7)$

0	3
1	
2	
3	4

 $PF = 6$ LRU

0	3	2
1		3
2		4

 $PF = 10$ $F = 4$:

0	4
1	
2	3
3	2

 $(10 > 8)$ $(8 < 10)$ FIFO

0	3	4
1		2
2		3

 $PF = 9$ $F = 4$:

0	3	2
1		1
2		3

 $PF = 10$ $Belady Anomaly.$

Here it is different

 $0 1 2 3 4 0 1 2 3 4$

65. STACK ALGORITHMS

⇒ The main reason for Belady Anomaly is "stack property."

⇒ optimal property is a stack algorithm

⇒ whenever a algo is a stack algorithm it does not follow Belady Anomaly

Optimal

RF = 0	1	2	3	0	1	4	0	1	2	3	4
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	2 0	3 3	3 3
1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1	1 1
2 2	2 2	3 1	3 2	3 2	4 2	4 2	4 2	4 2	4 2	4 2	4 2
3 3	3 3	3 3	3 3	3 4	3 4	3 4	3 4	3 4	4 4	4 4	4 4

when $f=3$ when $f=4$.

pages(3) \leq pages(4)

[pages(m) \leq pages(m+1)] → If an algo follows this property

then it is called Stack Algorithm
and it does not follow Belady Anomaly

Anomaly

FIFO Algorithm

RF =

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

frame=3
frame=4

Here pages(3) \neq pages(4)

[pages(m) \neq pages(m+1)] ⇒ It is not stack algorithm
and so it follows Belady anomaly

66. Q4 QUESTION ON LRU, LFU AND FIFO

A memory page containing a heavily used variable that was initialized very early and is in constant use is removed when _____ is used.

- a) LRU
- b) FIFO ✓
- c) LFU
- d) None

heavily used = frequently used

very early = one of the oldest pages which is available

constant use = Recently used page

68. GAI

A system
no page
some o
faults c

2196 b)

1

100

page
decre
becau
preser

69. SAME

⇒ In case

⇒ Increase

RS = 1 2

Optimal =

1	X 3
2	X 4
3	X 5
4	X 6

12PF

RS = 1 2

⇒ optimal

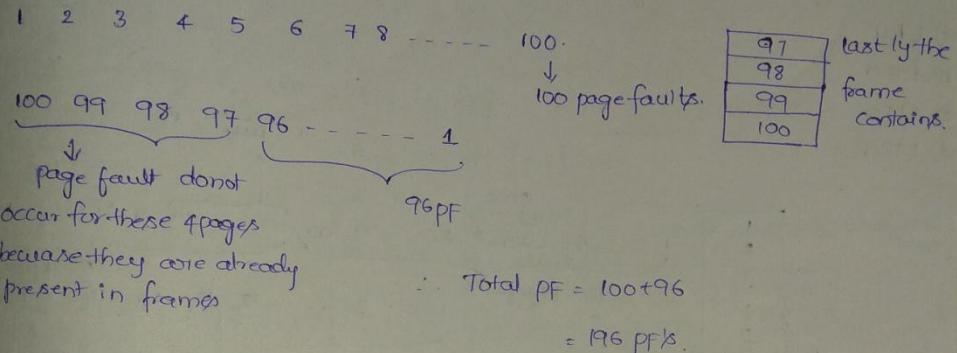
⇒ MRU

→ It is v

(92) 68. GATE 2010 QUESTION ON FIFO

A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first access 100 distinct pages in some order and then access the same in reverse order. How many page faults occur?

- a) 196 b) 192 c) 197 d) 195



(93) 69. SOME INTERESTING BEHAVIOUR OF OPTIMAL PAGE REPLACEMENT

→ In case of loops optimal and MRU (Most Recently used) give same

→ In case of loops both LRU and FIFO behaves in worst manner (more PF)

RS = 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6

optimal = 12 PF MRU = 12 PF LRU = 24 PF FIFO = 24 PF

X	Y	Z
X	Z	Y
Z	Y	X
Y	X	Z

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

Y	X	Z	1
Z	Y	X	2
X	Z	Y	3
Y	X	Z	4
Z	Y	X	5

X	Y	Z	1
Z	X	Y	2
X	Z	Y	3
Y	X	Z	4
Z	Y	X	5

12PF

24PF

24PF

RS = 1 2 3 4 5 6 7 8 9 | 9 8 7 6 5 4 3 2 1

→ Optimal behaves as LRU / FIFO and both give same PF's

→ Optimal uses a single page frame and it gives worst case

→ It is very difficult to implement optimal practically.

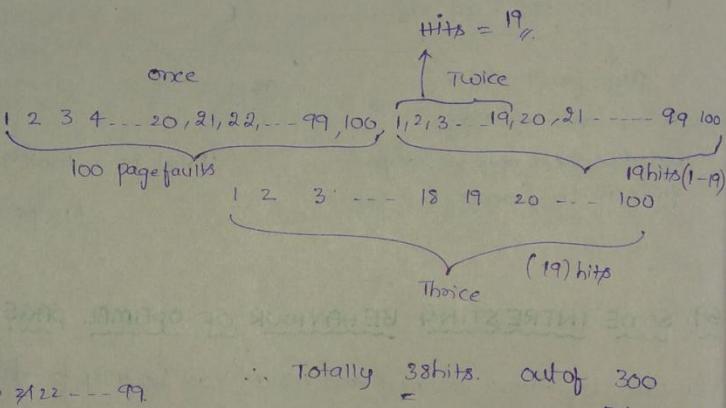
To: GATE 2014 QUESTION ON LRU, MRU, LIFO, FIFO AND OPTIMAL

A4

A computer has 20 physical page frames which contains pages numbered 101 through 120. Now a program access the pages numbered 1, 2, ..., 100 in that order and repeats the access sequence thrice. Which one of the following page replacement policies experience the same no. of page faults as optimal page replacement policy for this program?

- a) LRU
- b) FIFO
- c) LIFO
- d) MRU

1	
2	
3	
4	
5	
18	
19	26 21 22 ... 99
20	21 22 23 ... 100



a) LRU.

101	X	21	41
102	F	22	42
103	S	23	
	!	!	
120	26	46	66

300 misses. (no hits at all)

b) Similarly 300 misses

c) Similarly also 300 misses. The actual Ques should be which of the following give same pattern of page faults. Then option 'd' will be the ans

II. WORKING SET

Initially
frames

→ This a

RS: 1 2

window

Now

w=

w=

w=

w=

w=

w=

Ex :

ade | c

w= fad, e

A=4

w= fad, e

w= fd, re

w= fd, re

7.1 WORKING SET ALGORITHM

bered
100 in
be following
optimal

NOTATION : (95)

Initially the process requires less frames but gradually the necessity of the frames increases.

⇒ This algo has the parameter "Windowsize (A)"

RS: 1 2 3 1 2 4 1 4 2 1 5 1 2 4 2 1

window Size A = 4.

Now working set when A=1 is {1}

A=2 is W = {1, 2}

A=3 is W = {1, 2, 3} working set size ≤ window size

A=4 is W = {1, 2, 3}

$W \subseteq A$

W = {1, 2, 3} A = [2 3 1 2]

W = {1, 2, 3, 4} A = [3 1 2 4]

W = {1, 2, 4} A = [1 2 4 1]

W = {1, 2, 4} A = [2 4 1 4]

W = {1, 2, 4} A = [4 1 4 2]

⇒ proposed to allocate frames

dynamically

⇒ very difficult to implement practically.

Ex :

a d e | c c d b c e c e a d

W = {a, d, e} = Initially window's Snapshot

A = 4

W = {f, a, d, e, c} = 5F

W = {d, e, r, c} = 4F

W = {f, d, e, r, c} = 5F

W = {c, b, d} = 3F

W = {c, b, d} = 3F

W = {d, r, b, c, e} = 5F

W = {b, r, e, c} = 4F

W = {b, r, e, c} = 4F

W = {c, r, e} = 3F

W = {f, a, r, c} = 4F

W = {f, a, r, c, d} = 5F

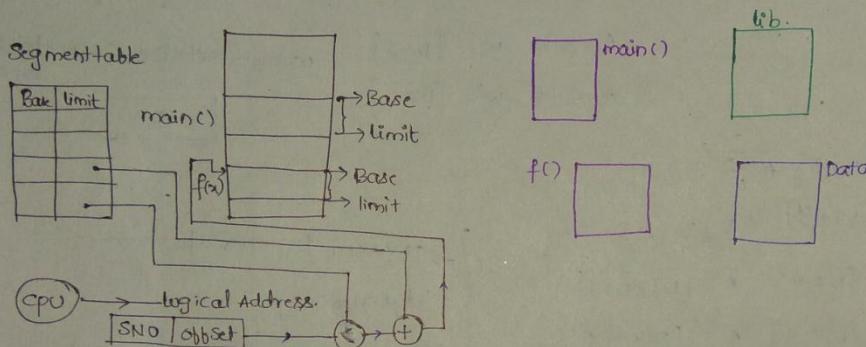
the following

ans

$$\text{Avg. Frame requirement} = \frac{4+3+3+3+4+3+2+3+4}{10} = 3.2$$

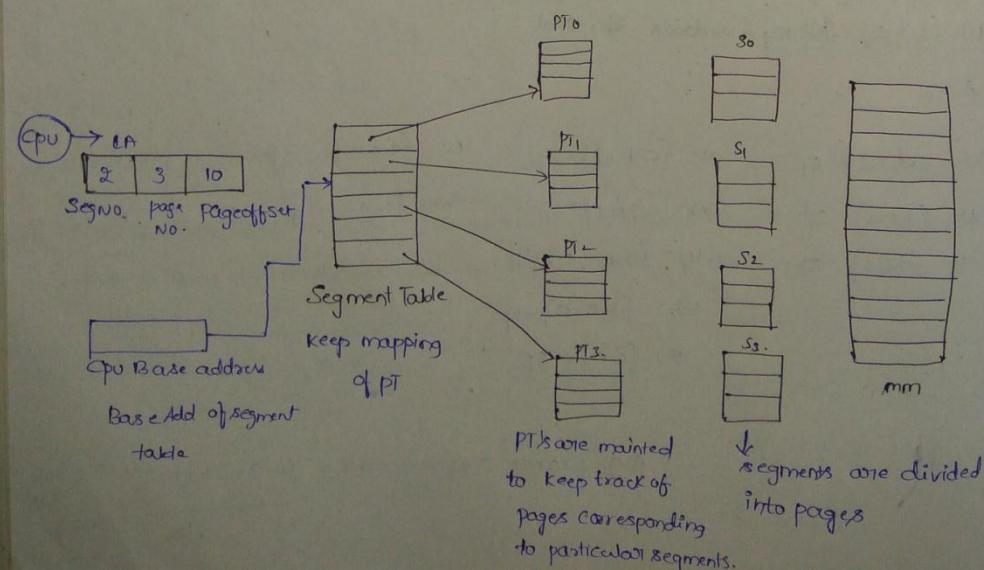
75. SEGMENTATION

- ⇒ When we do paging some useful instructions of the program may fall in different pages which are supposed to be together. So segmentation is used.
- ⇒ Segmentation gives the impact of division as how we assume the division should be.
- ⇒ The program is divided into no. of segments and each segment is loaded in main memory. It has Base and Limit Registers that says the starting address of a particular segment and end of a particular segment.

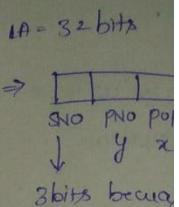


⇒ offset should always be less than limit

76. SEGMENTED PAGING



77. GATE 91



Now, PTS

2^2

$\Rightarrow 2^2$

17. GATE 99 QUESTION ON SEGMENTATION PAGING

IA = 32 bits

$$\Rightarrow \boxed{\begin{array}{|c|c|c|} \hline S & N & O \\ \hline P & N & O \\ \hline F & P & O \\ \hline \end{array}} = 16 \text{ bits}$$

↓ ↓ ↓
y x

3 bits because 8 segments are present $\Rightarrow 3+x+y=16$

$$\Rightarrow \boxed{x+y=13}$$

Now, PTS = No. of pages * (PTE)

$$2^x = 2^y * 2$$

$$\Rightarrow 2^x = 2^{y+1} \Rightarrow x=y+1$$

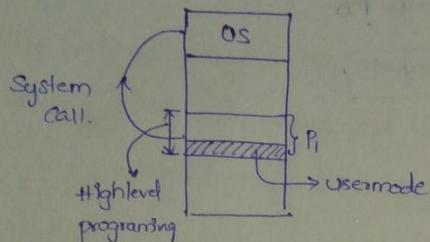
$$\Rightarrow x+y=13$$

$$\Rightarrow y+1+y=13 \Rightarrow \boxed{y=6, x=7} \quad \therefore \text{Size of page} = 2^x = 2^7 = 128B.$$

6. THREAD AND SYSTEM CALLS

1. SYSTEM CALLS VS FUNCTION CALLS

⇒ System call is same as the function call but then in system call we will be trying to call the function which is present in the operating system.



```
printf()
{
    write()
}
```

7. FORK

⇒ fork()

⇒ when

⇒ when
and

⇒ fork
fork

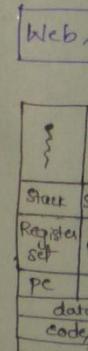
2. PROCESS CONTROL SYSTEM CALLS

Process control

- End, Abort
- Load, Execute
- Create process, Terminate process
- Get process attributes, Set process attributes
- wait for time
- wait event, Signal event
- Allocate, and Free memory.

System calls required to control a process.

8. PROCESS



Thread: Ne

blo

3. FILE MANIPULATION

- Create file, delete file
- Open, close
- Read, write, reposition
- Get file attributes, (Set file attributes.) if you are owner of file you will be provided to manipulate the file
 - ↓ (Date of modification), (owner of file), size of file,
 - (Date of creation), (permissions)

9. USER

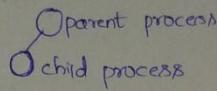
→ The dis

Kernel level

7. FORK SYSTEM CALL

→ `FORK()` / `execve()` is the Systemcall

↳ when any process calls `fork()` system call it will create a child process

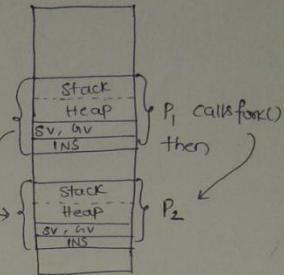


→ when `fork()` is called there should be Context switch and we should shift from Usermode → Systemmode

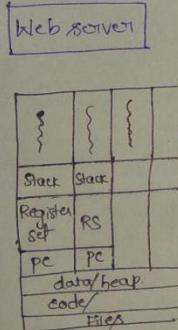
Copied Overhead.

→ `fork` returns `i=0` for child process.

`fork` returns `pid of child` for parent process } `fork()` creates two process (one is already existing one and new one)



8. PROCESS VS THREADS



process (<code>fork()</code>)	Thread (Userlevel)
→ System calls are involved	→ NO Systemcall
→ Context switch	→ Register set switch
→ diff copies of code and data	→ same copies of code and data

Thread: New, Ready, Running,

Blocked, Terminated

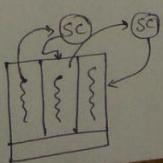
9. USER LEVEL V-S KERNEL LEVEL THREADS

→ The disadvantages of user level threads is

→ Blocking systemcall will block the whole task.

→ unfair scheduling

KERNEL LEVEL THREADS



- parallelism is guaranteed.
- DisAdv: Expensive compared to UI
- switching requires more time (requires Systemcall for switching)