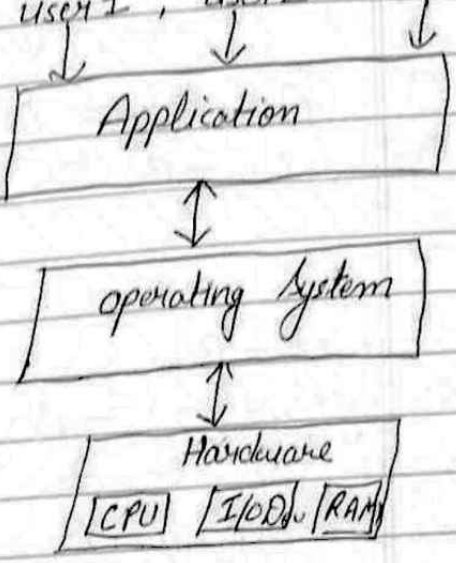
Operating System

Operating System is a System Software which acts as a intermediate b/w user & hardware



if we don't have any operating system than we have to write programs for each and every task

Functions of operating System

1. Resource management
2. Process management (CPU Scheduling)
3. Storage management (H.D) → file System
4. Memory management (RAM)
5. Security

↓

Windows uses kerberos Security Protocal which uses the password to provide safe access of the file & the user.

when any process calls or executes an instruction outside its Segment or block, then the process on the whole gets totally blocked.

The privacy & Security is maintained even among the processes so that they don't interfere or interchange data with each other.

P. – P10 →

Matrices

ARFAT

Types of operating System

① Batch — Punch cards, Paper tape, magnetic tap → operator

Batches: B_1 B_2 B_3

IBM → Fortran → IB SYS 709x → 1960

3 5 7 → CPU → Idle

CPU → I/O

• Multiprogrammed OS → Non-Preemptive

↓ Idleness R.A.M

In this type C.P.U exed executes the process one by one completely & than turn on to next process. Suppose we have 4 processes to execute P_1, P_2, P_3, P_4. first it executes the P_1 completely & than turn on P_2 then P_3 & so on. but if process P_1 said that I have to go for input output instruction if it happens in that case C.P.U turn on to process P_2.

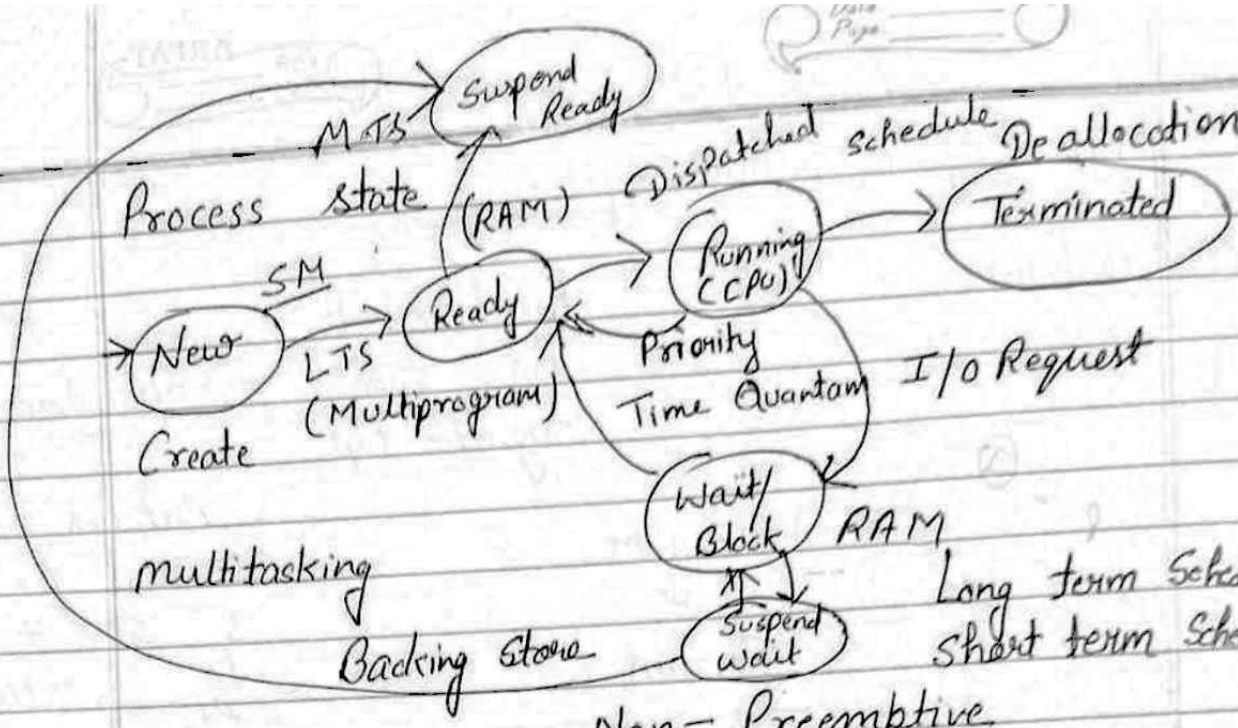
Multitasking → Time sharing → Response time

↓ Responsivness

Distributed →

Cluster →

Embedded →



when a process executes it passes through different states. This stages may differ in different operating system.

Process Scheduling Algorithm

Pre - Emptive	Non Pre - Emptive
SRTF (Shortest Remaining time first	FCFS (First come First Serve)
	SJF (Shortest job First)
LRTF (longest Remaining time first	LJF (longest job first)
	HRRN (Highest Response Ratio Next)
Round Robine	
Priority end	Multilevel Queue Priority

C.P.U Scheduling

Arrival time :- The time at which process enters the ready Queue or state.

Duration

Burst time :- Time required by a process to get executed by C.P.U.

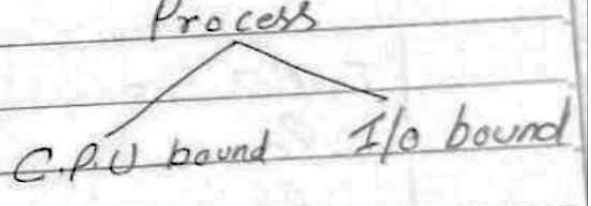
Completion time :- The time at which process Complete its execution.

Turn Around Time :- { Complete time − Arrival time }

Waiting time :- { Turn Around time − Burst time }

Response time :- { (The time at which a process get CPU first time) − (Arrival time)

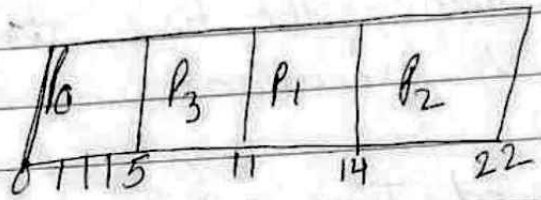
2^{x3}



Process	AT	B.T
P_1	1	8 7 5
P_2	3	2 3
P_3	6	1 0 2
P_4	7	8 10
P_5	9	8

Priority Scheduling

Priority	AT	BT	Priority
P_0	0	5	1
P_1	1	3	2
P_2	2	8	1
P_3	3	6	3



W.T = R.T - A.T

P_0 = 0 - 0 = 0

P_1 = 11 - 1 = 10

P_2 = 14 - 2 = 12

P_3 = 5 - 3 = 2

T.A.T = C T - A T

P_0 = 5 - 0 = 5

P_1 = 14 - 1 = 13

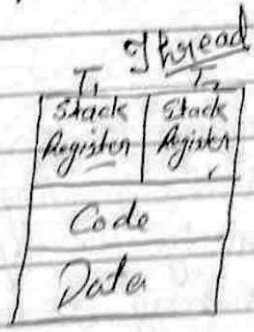
P_2 = 22 - 20 = 20

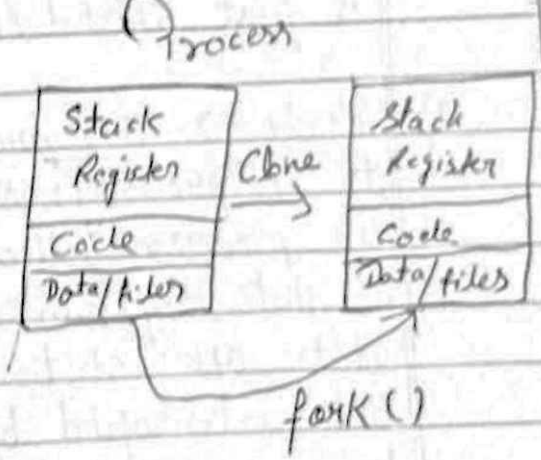
P_3 = 11 - 3 = 8

Round Robin

Process & Thread

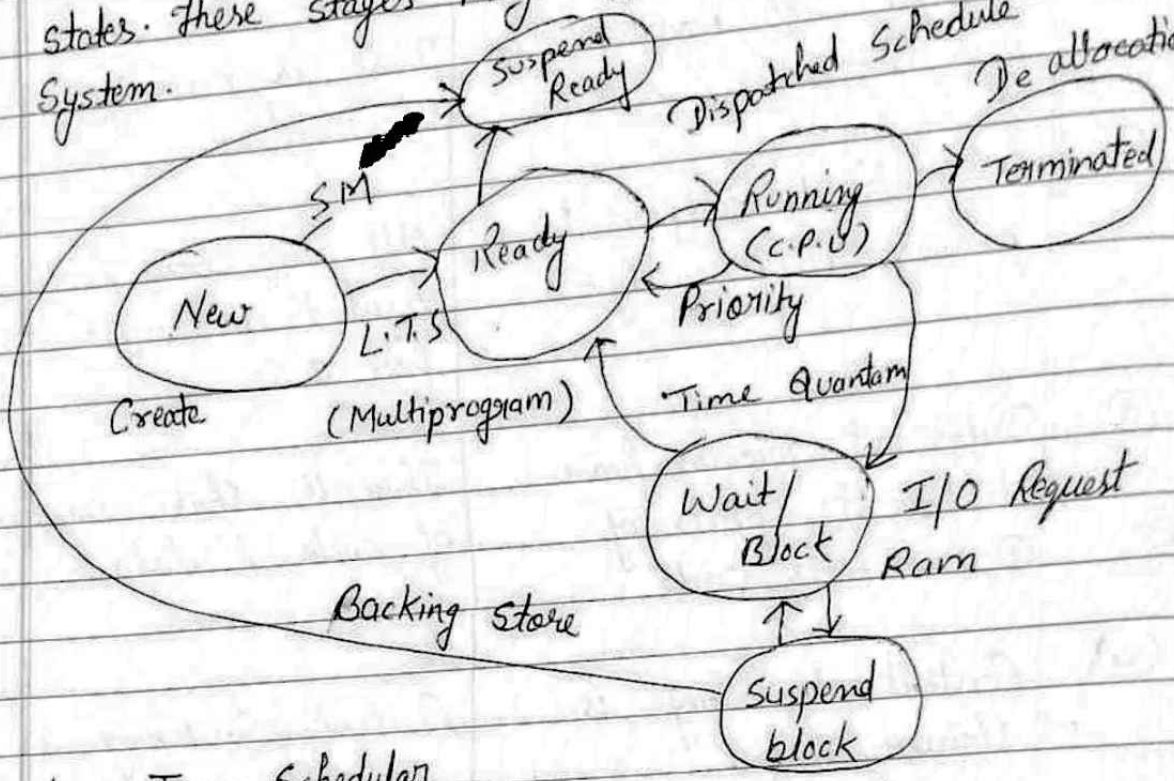
	Process	Thread
①	System call involved in process.	There is no system call involved.
②	O.S treats different process differently.	All user level threads treated as single task for O.S.
③	Different process have different Copies of Data, files, code.	Threads share same Copy of code & data.
④	Context Switching is slower	Context Switching is faster
⑤	Blocking a process will not block another	Blocking a thread will block entire process
6	Independent	Interdependent





Process state

when a process executes, it passes through different states. these stages may differ in different operating System.



Long Term Schedular — Non-Preemptive
Short Term Schedular — Preemptive
Mid Term Schedular

1) Start → This is initial state when a process is first started / created.

2) Ready → The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating System so that they can run. Process may come into this state after start state or while running it by but interrupted by the schedular to assign C.P.U to some other process.

3.) Running - After Ready state, the process state is set to running and the processor executes its exe- instruction.

4.) Waiting :- Process moves into the waiting state if it needs to wait for a resources, such as waiting for user input, or waiting for a file to become available.

5.) Terminated or Exit :- Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it is waits to be removed from main memory.

① what is process ?

Process is a program in execution including the current values the program counter, Register, & variables. The difference b/w a process the programmers that the program is a group of instruction where is the process is the activity.

we write our program in a text file & when we execute this program, it becomes a process which performs all the task mentioned in the program.

when a program is loaded into the memory and it becomes a process, it can be divide into 4 section - stack, heap, text & data.

1) Stack :- The process that contains the temporary data such as Method/function, parameters, return address & local variables.

2) Heap :- This is dynamically allocated memory to a process during its runtime.

3) Text - This includes the current activity represented by the value of program counter & contents of the processor's Register.

4) Data :- This section contains the all global & static variables.

- Process Table & Process Control Block (PCB):

Q What is process table?

The operating system manages and control the resources by having a table. Tables are important data structures to store information about every process & resources.

This is the reason the operating system maintain memory tables, input output tables & process table.

The process tables store the ID of every process & corresponding to it, the pointer to its PCB

1.	Running	•
2.	Ready	•
3	Blocked	•
4	Suspend	•

Pointers to PCB

Q What is process control Block ? :

A process control block is a data structure maintained by the operating system for ever process.

Some operating system maintain only PCB in that PCB has all entries stored by a process table

A PCB keeps all the information needed to keep track of a process.

Process ID
State
Pointer
Priority
Program counter
CPU Register
I/O Information
.
.
etc

① Process state :- Current state of the process i.e, ready, running, or waiting etc.

② Process Privileges :- allow/ disallow access to the system resources.

③ Process Id :- Unique identification of each of the process in the operating system.

④ Pointer :- A pointer to parent process.

⑤ Program Counter :- Program counter is a pointer to the address of the next instruction to be executed for this process.

⑥ CPU Register :- Various CPU register where process need to be stored for execution for running state.

⑦ C.P.U Scheduling Information

⑧ Memory Management Information :- Information of page table, memory limit, Segment table information.

⑨ Accounting Information :- It includes the amount of CPU used for process execution, time limits, execution ID etc.

⑩ Input Output Status Information :- This includes list of input output devices allocated to the process.

The PCB is maintained for a process throughout its lifetime, And is deleted once the process

terminates :

Overview of Inter - Process Communication

IPC = (Inter - Process Communication) is the capability which allow two process of the computer to communicate with each other.

where will be those process?

Its not important for process to be in the same computer they can be in same computer or in different computer which are connected over the network.

IPC allow one application to control other application, it also allow two applications to share data with each other.

IPC is an important thing in the multiprogramming system, or we can say multiprogramming system depends on this system.

Single process operating system do not use IPC.

Microsoft windows is a Multiprogramming operating system & it use the IPC. In windows IPC is known as Dynamic Data Exchange System.

In an OS (Multiprogramming OS) many process may be running at the same time. In the context of IPC we can divide these processes into two main categories.

1. Independent Process
2. Cooperating Process

Independent Process

As name shows in the multiprogramming OS environment these processes are work independently.

They never ask any other process to share data & never share their own data to some other process.

Although they are in the Multiprogramming OS but they never share data.

Cooperating Process

Name is telling the whole store these are just opposite to the independent process & share the data with other cooperating process into the System.

Different Advantages of process communication

(1) Information sharing
(2) Computation Speed up
(3) Modularity
(4) Convenience

Computation speed up

Task get divided into multiple task, all task run parallel & achieve the required result.

Modularity

Divide a function into sepra separate process & threads.

Convenience

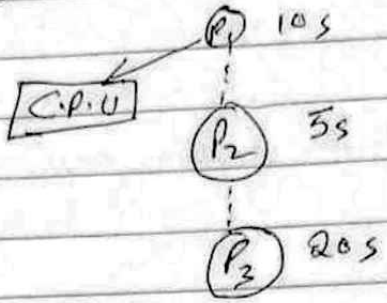
If user is using the same data in different task conflict can be arise so this system try to avoid conflicts.

There are further two memory models in this system.

Shared Memory Model
Message passing System

FCFS (First come first serve) C.P.U Scheduling

C.P.U Scheduling is a technique in operating system which handles the removal or selection of processes.



Types

Non-Preempite	Preemptive
① Once a process enters in C.P.U until it terminates other processes have to wait.	① if a low priority task is currently in the C.P.U and another task with high priority demand C.P.U access. First priority will be removed until High priority task terminated
② Priority is not considered	

FCFS Scheduling Algorithm

- Works on first come first serve basis.
- It is Non-preemptive

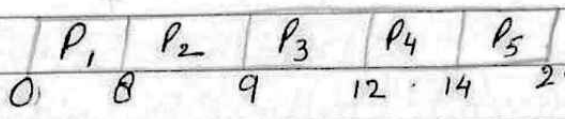
Ques Consider following table & calculate Average TAT & Average waiting time using FCFS Algorithm.

P_1 - Execute

Process	Arrival time	Burst time	Priority
P_1	0	8	3
P_2	1	1	1
P_3	2	3	2
P_4	3	2	3
P_5	4	6	4

Ans

Gantt chart



P No.	AT	BT	C.T	T.A.T (C.T - A.T)	WT (TAT - BT)
P_1	0	8	8	8	0
P_2	1	1	9	8	7
P_3	2	3	12	10	7
P_4	3	2	14	11	9
P_5	4	6	20	16	10

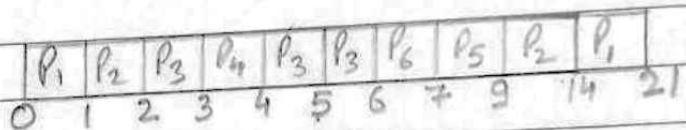
Average W.T $= \frac{0+7+7+9+10}{5} = \frac{33}{5} = 6.6$

Average TAT $= \frac{8+8+10+11+16}{5} = \frac{53}{5} = 10.6$

SRTF / SJF (Preemptive)
Shortest remaining time first

P.No.	A.T	B.T	C.T	T.A.T (CT-AT)	W.T (TAT-BT)
P_1	0	8	21	21	13
P_2	1	6	14	13	7
P_3	2	3	6	4	1
P_4	3	1	4	1	0
P_5	4	2	9	5	3
P_6	5	1	7	2	1

Gantt chart



Average T.A.T = $\frac{45}{6} = 7.5$

Average W.T = $\frac{25}{6} = 4.16$

SJF (Non-Preemptive)
(Shortest Job first)

Process No	A.T	Burst time	C.T	T.A.T	W.T
P_1	0	6	6	6	0
P_2	2	1	7	5	4
P_3	4	4	14	10	6
P_4	5	3	10	5	2

Gantt chart

Average T.A.T = $\frac{26}{4}$ = 6.5

Average W.T = $\frac{12}{4}$ = 3

Priority Process ~~Scheduling~~
Round Robin Algorithm (Preemptive)

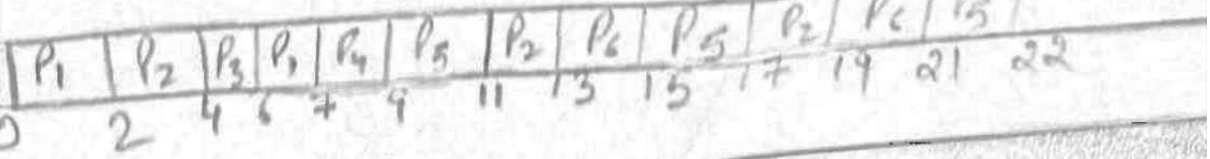
Round Robin

Time Quantum : 2 units.

Process Number	Arrival Time	Burst time
P_1	0	3 1 0
P_2	1	6 4 2 0
P_3	2	7 5 0
P_4	3	2 0
P_5	4	5 3 1 0
P_6	5	4 2 0

Queue | P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12}

Gantt chart

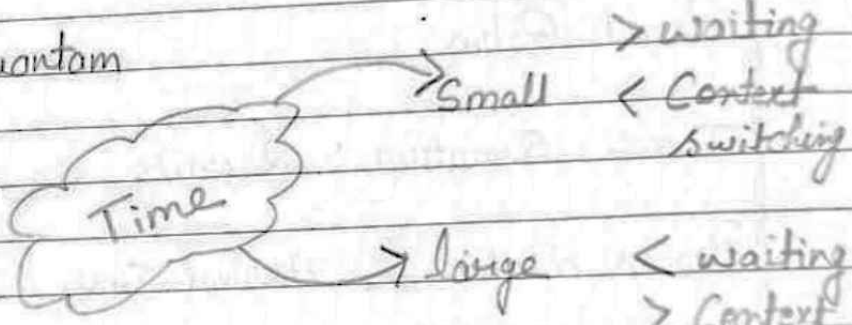


Process Number	A.T	B.T	C.T	TA.T	W.T
1	0	3	7	7	4
2	1	6	19	18	12
3	2	2	6	4	2
4	3	2	9	6	4
5	4	4	22	18	14
6	6	4	21	15	11

Average T.A-T $= \frac{7+18+4+6+18+15}{6} = 11.33$

Average W.T $= \frac{4+12+2+4+14+11}{6} = 7.83$

Time Quantam



Priority Process Scheduling

Process No.	A.T	B.T	Priority
P_1	0	3	2
P_2	1	10	4
P_3	2	7	6
P_4	3	3	10
P_5	4	4	8
P_6	5	4	12
P_7	6	6	9

Gantt chart

P_1	P_2	P_3	P_4	P_6	P_4	P_7	P_5	P_3	P_2	P_1
0	1	2	3	5	9	12	18	22	29	30	33

	C.T	T.A.T	W.T
1	33	33	29
2	30	29	27
3	29	27	19
4	12	9	4
5	22	18	14
6	9	4	0
7	18	12	6

Average T.A.T $= \frac{33+29+27+9+18+4+12}{7} = 188$

Average W.T $= \frac{29+27+19+4+14+0+6}{7} = 14.14$

Process Synchronization

Process

Cooperative Process

Execution of one process affects the execution of other process

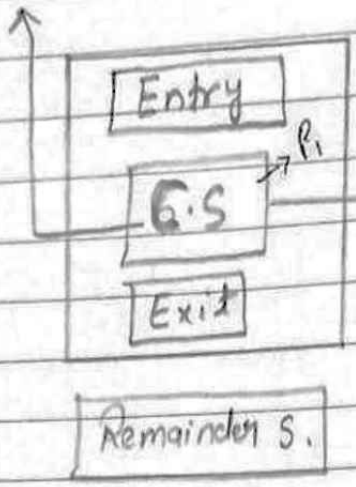
↓ variable
Code
Resources

independent process

↓

execution of one process does not affects the execution of other

Critical Section Problem



Mutual Exclusion
↳ only one can access

Portion of program where Shared variable, shared data, shared resources placed.

Process may change
→ Common variable
→ update table

Two process can have Entry & Exit & Remainder Section but in critical section only one process can occur.

How do C.P.U known that one problem is in the section we do not send the other one?

The solution of critical section must satisfy following requirements.

1) Mutual Exclusion: under no circumstances can two process be in their critical section for the same variable at the same time.

2) Progress:- No process remain forever in C.S. should be in progress of process.

3. Bounded waiting :- Bound on number that times

process should have to wait forever for C.S.

Solution :- S/w Type :- lock variable, strict Alternation peterson ago
H/w type :- T.S.L / Test & set
O.S type :- Semaphore
Compiler type :- Monitors

Semaphores

Semaphores was proposed by Dijkstra is 1965. Semaphores is simply a variable which is non-negative & shared between Threads. This variable is used to solve the critical section problem & to achieve process synchronization in the multiprocessing environment.

Semaphores are of two types :-

1. Binary Semaphore -> In
   :- This is also known as mutex lock. It can have only two values - 0 & 1. Its value is initialized to 1. It is used to implement the solution of critical section problem with multiple processes.
2. Counting Semaphore :- It's value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

Process Synchronization

In this task the execution of processes in such a way that no two processes can have access to the same shared data & resources. It is a procedure that is involved in order to preserve the appropriate order of execution of Cooperative processes.

Deadlock

when any process P_1, P_2, P_3 wanted to access single resource R, Than there deadlock occur. Because all the process wanted to use it.

Necessary & Sufficient conditions for Deadlock

1) Mutual Exclusion:-

Resources → Not sharable

P_1 → resource

Only one can hold at a time after release another one can

2 Hold & wait

process holding a resources & waiting for additional resources

↓

held by another process

3. No preemptive Condition:- Resources cannot be removed from the processes are used to Completion or released voluntarily by the process holding it.

Priority Ex. :- higer priority
lower priority
preempted → C.P.U

Circular wait :- Each process wait for next process in circular manner.

Deadlock Prevention :-
-> One of the necessary condition can never hold

① Mutual Exclusion :- Not always possible to prevent deadlock by preventing M.E
(Making all Resources Shanable)

② Hold & wait :-
before wait :- all required resources occured.
Reduce throughput :- those not needed even get earlier.

All required resources might not be free always.

③ No preemption :-
Resources can be preempted.
request → resource { all other resources which are holds preempted
↑
held by another waiting resource.

④ Circular wait :- Resources may be ordered (increasing order)

Deadlock Avoidance

How resources are to be requested.

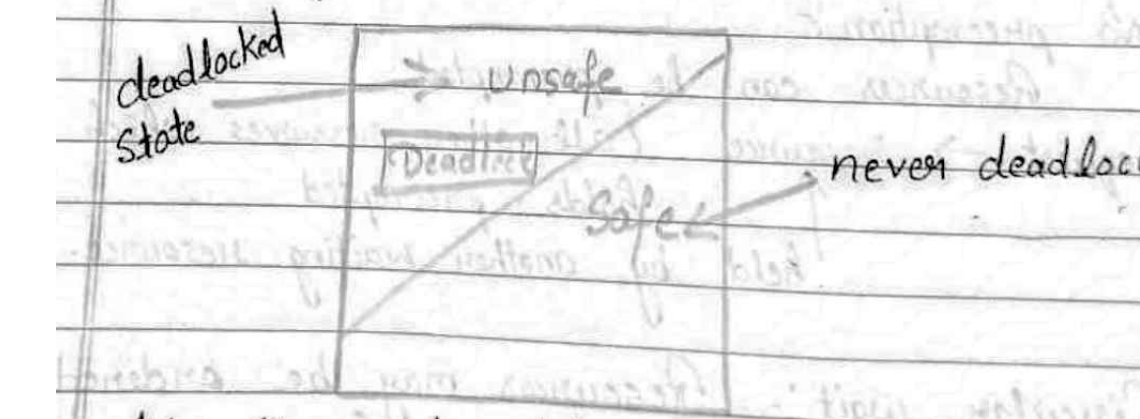
Currently available, the resources currently allocated to each process & the future requests & releases of each process.

Examine → Resource allocation
Circular → wait can never exist.

Resource allocation state
↓
no. of available resources maximum demands of the processes.

Safe - State :-
Safe Sequence :→ allocated to peek of resource

No Deadlock
$< P_1, P_2, \ldots, P_n >$ for each P_i satisfied by the currently.



Not all unsafe states are deadlocks.
behaviour of the processes controls unsafe states.
To avoid deadlock → Safe State (always)

Deadlock Recovery

How to recover from deadlocks?

Killing Process

Killing all process involved in deadlocks.
Killing process one by one.
After killing each process check for deadlock again } ← Repeat steps.

Resource Preempition.

Resources are preempted from the processes involved in deadlocks, preempted resources are allocated to other processes, so that their is a possibility of recovering the system from deadlock ← Starvation

Deadlock Detection

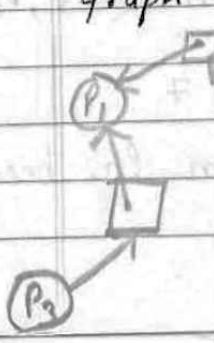
D.D

Wait for graph

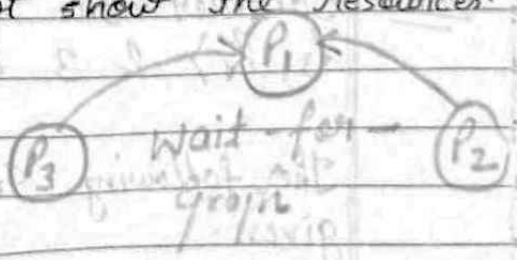
Bankers Algorithm

Wait for graph

It is an enhanced version of Resource allocation Graph in which we do not show the resources.



Resource Allocation Graph



Wait for graph

If all the resources have single instances & cycle is being formed, then the system is in Deadlock. If cycle is there, but resources have more than 1 instance, then the deadlock may or may not be present.

Bankers Algorithm

Bankers algorithm is a deadlock avoidance algorithm. It is named so because this algorithm is used in banking system to determine whether a loan can be granted or not.

Every time a loan has to be granted by the bank. it subtracts the loan amount from the total money the bank has. Then it checks if that difference is greater than. It is done because only then, the bank would have enough money even if all the account holders draw all their money at once.

Bankers algorithm works similar in Computer

whenever a new process is created. it must specify the maximum instances of each resource type that it needs, exactly.

Ques Consider the system with 5 process $\langle P_0, P_1, P_2, P_3, P_4 \rangle$ & 3 Resources types A B C
10 5 7

The following Snapshot of the System has been given.

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2	5	3	2
P_2	3	0	2	9	0	2	7	4	3
P_3	2	1	1	2	2	2	7	4	5
P_4	0	0	2	4	3	3	7	5	5

a) find need matrix

b) is the system in a safe state if yes find safe Sequence.

① find need matrix

need matrix = (max − Allocation)

	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Safety Algorithm

if

need ≤ available

then

Execute Process

new available = Available + Allocation

Else

Do not Execute go forward

P_0 → need > available
7, 4, 3 > 3, 3, 2
do not execute P_0

P_1 → need ≤ available
1, 1, 2 ≤ 3, 3, 2
Execute P_1
new Available = Available + Allocation
= 3, 3, 2 + 2, 0, 0
= 5, 3, 2

P_2 → need > available
= 6, 0, 0 > 5, 3, 2
= do not execute P_2

P_3 → need ≤ available
0, 1, 1 ≤ 5, 3, 2
Execute P_3
new available = 5, 3, 2 + 2, 1, 1
= 7, 4, 3

P_4 → need ≤ available
= 4, 3, 1 ≤ 7, 4, 3
Execute P_4
new available = 7, 4, 3 + 0, 0, 2
= 7, 4, 5

P_0 → need ≤ available
7, 4, 3 ≤ 7, 4, 5
Execute P_0
new available = 7, 4, 5 + 0, 1, 0
= 7, 5, 5

$P_2 \rightarrow$ need $\leq$ available

$6, 0, 0 \leq 7, 5, 5$

Execute P_2

new available $= 7, 5, 5 + 3, 0, 2$

$= 10, 5, 7$

Safe Sequence will be

$< P_1, P_3, P_4, P_0, P_2 >$ $P_1 \rightarrow R_1$

Memory management

Memory → Storage

written in the form of $2^n \times m$

n = # Address lines

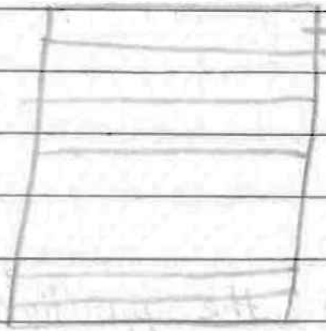
m = # datalines

A·L = Capacity

D·L = Size of data

Byte Addressable

Bit Addressable



Always starts with 0.

$2^n \times m$

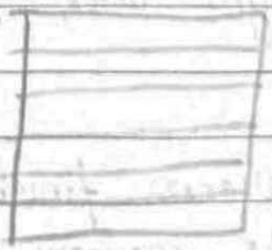
$2^3 \times 1$

Ex - Memory addressing in Memory

① $8 = 2^3 \times 1$

0 - 7

① $16 = 2^4 \times 1$

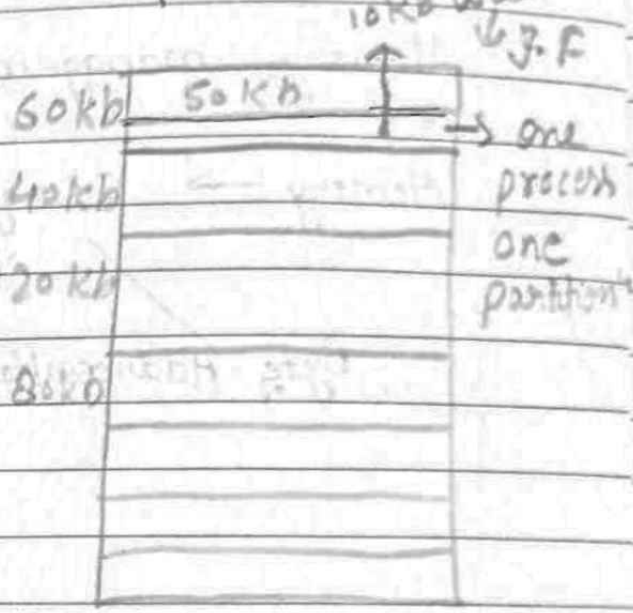


0 - 15

Memory Management Techniques

- Continuous
  - ↳ Fixed - Partition Scheme
  - ↳ Variable " "
    - ↳ First fit
    - ↳ Best fit
    - ↳ Worst fit
    - ↳ Next fit

Non - Continuous
- Paging
- → Multilevel paging
- → Inverted Paging
- Segmentation



fixed partition Scheme

No. of partition : fixed

Size of partition : Not fixed

One block holds only one partition

wastage of memory inside the partition
↳ Internal fragmentation

Variable Partitioning Technique

whatever / whenever process block comes in accordingly it assigns block. (at the time partition will be made)

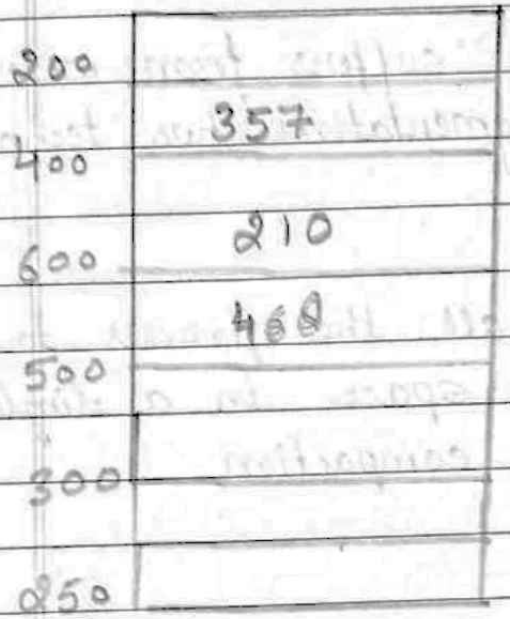
When process finish its execution it move out from the memory and a free partition / hold will

be made.

Decision making for accomodation of partition fit Algo use.

① first fit

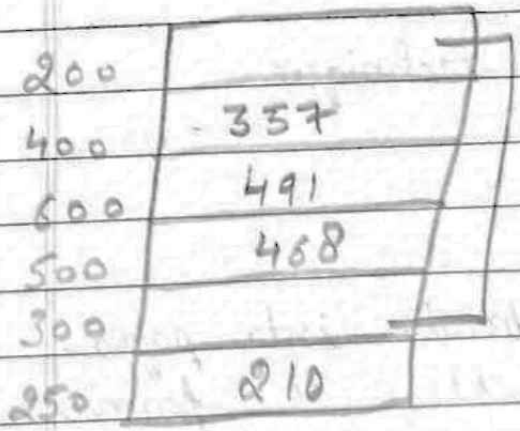
Ques → 357, 210, 468, 491
→ Partition :- 200, 400, 600, 500, 300, 250



In this method, first job claims the first available memory with space more than or equal to it's size. The operating system doesn't search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

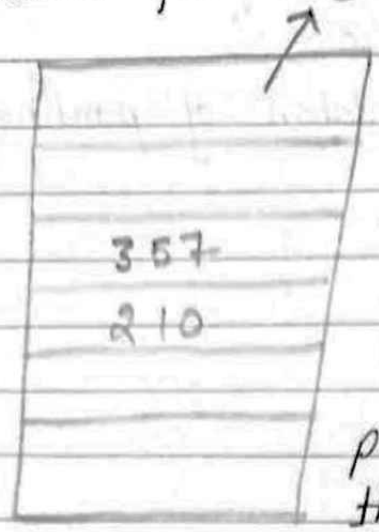
Remain → 491

② Best fit



The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. It then tries to find a hole which is close to actual process size needed.

worst fit external fragmentation



worst fit allocates a process to the partition which is largest sufficient among the freely available partitions available in the main memory. if a large process comes at a later stage then memory will not have space to accommodate it.

Variable partition scheme suffers from external fragmentation. To avoid fragmentation two technique will be used.

Sol:

Compaction :- To move all the process towards the top / down & free space in a single continuous placed called compaction.

↳ disturb / interrupt all running process in the memory.

Implement non continuous technique

Non Continuous Scheme

1) Paging :- logical memory breaks into pages.
physical memory " " frames.

page size = frame size

Page Table

Pages

Page 0
Page 1
Page 2
Page 3

Page no. as Index

0	7
1	1
2	9
3	5

→ frame no.

Logical memory

Page table (mapping)
⇒ Page no. → Index

frame no. → base address of each page in physical memory.

frames

1	page 1	
2		
3		
4		
5	Page 5	
6		
7	Page 0	7
8		
9	Page	9

Logical Address space → L.A.S (byte / word)

Physical Address space → P.A.S (byte / word)

Logical Address (L.A) (bits)

Physical Address (P.A) bits

$L.A = 32$ bits

$L.A.S = 2^{32} = 4\ G.W$

$P.A = 28$ bits

$P.A.S = 2^{28}$

$= 256$

$L.A.S = 128\ M.W$

$= 2^7 . 2^{20}$

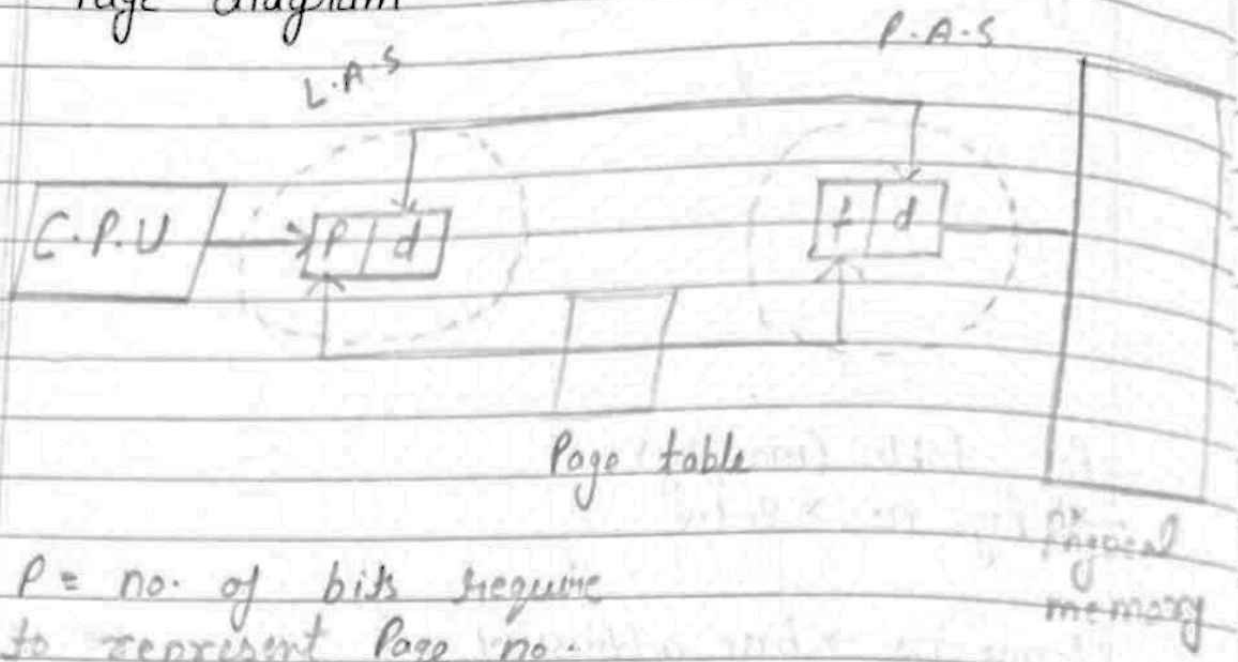
$L.A = 27$ bits

$P.A.S = 32\ M.W$

$= 2^5 . 2^{20}$

$P.A = 25$ bits

Page diagram



p = no. of bits require to represent Page no.
d = page offset

Technique of mapping processor generated address to physical Address is called Paging

$$\#\ Pages = \frac{L \cdot A \cdot S}{Page\ Size / f \cdot s} \qquad \#\ frame = \frac{P \cdot A \cdot S}{P \cdot S / frame\ Size}$$

Ex :-

System where L·A = 13 bits
P·A = 12 bits
Page Size = 1 K·W

L·A = 13 bit
L·A·S = 2^{13} W

P·A = 12 bit
P·A·S = 2^{12} W

No. of pages ?

No. of frame = ?

$$\text{No. of pages} = \frac{L \cdot A \cdot S}{\text{Page Size}} = \frac{2^{13}}{2^{10}} = 2^3 = 8 \text{ pages}$$

No. of frames = $\frac{P.A.S}{P.S / \text{frame size}}$

$= \frac{2^{12}}{2^{10}}$

$= 2^2$

$= 4$ frames

Performance of paging

Page table placed in main memory

Effective Memory Access time (EMAT) = M + M = 2M

(M ← Main memory Access time)

(one M → Page table reference

Another M → Actual memory reference

TLB (Translation Look a Side Buffer)

→ To improve performance TLB is added just before page table.

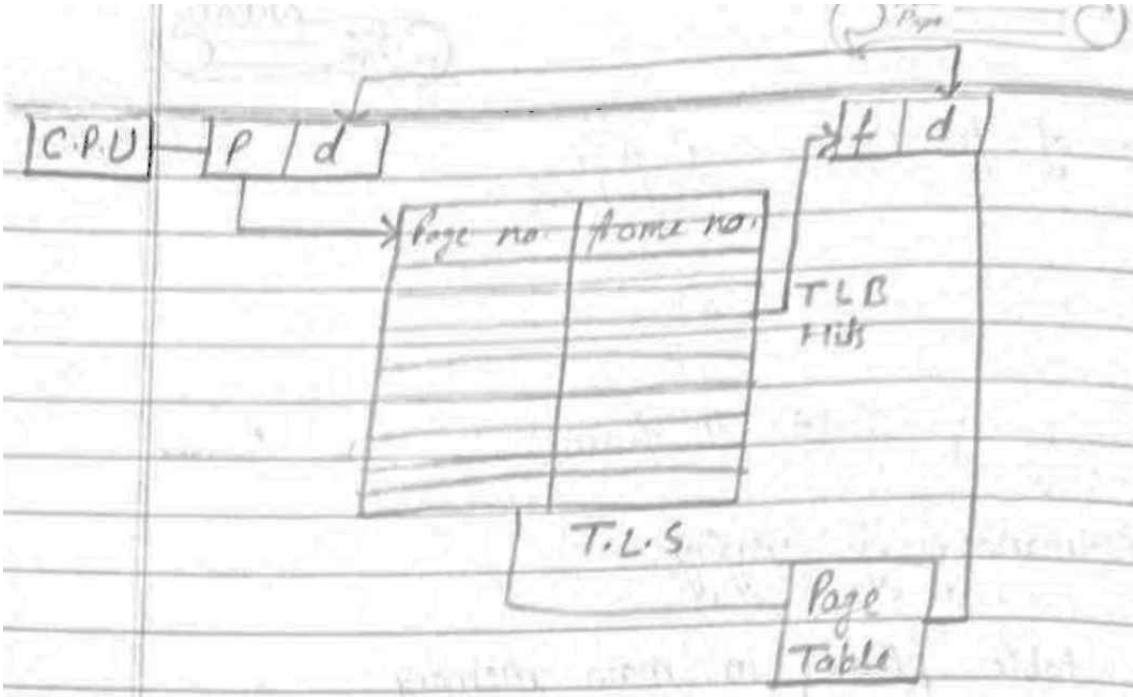
T.L.B contain frequently referred page no. and frame no.

if

reference found = TLB Hits

else

T.L.B Miss



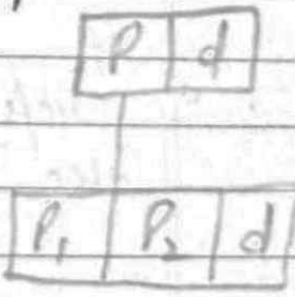
T.L.B Access time = C
Hit ratio = H

With T.L.B
EMAT = H [C+M] + [1-H] [C+2M]

Without T.L.B
EMAT = [illegible] 2M

Multilevel Paging

To avoid the overhead of maintaining the large page table → multilevel page is implemented.



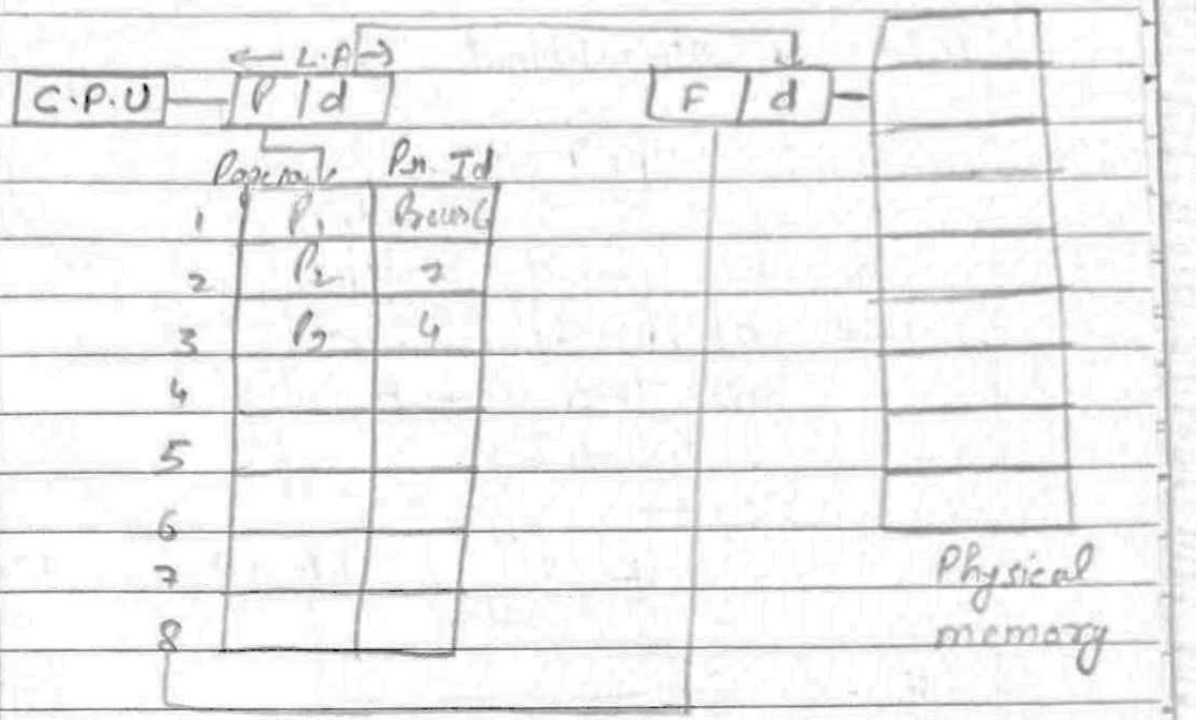
p is divided into two parts ⇒ p_1 & p_2

p_1 : outer table / no. of bit required of represent page of page table

p_2 : inner table / no. of bit for page size

Inverted Paging :- Only one table for all the process.

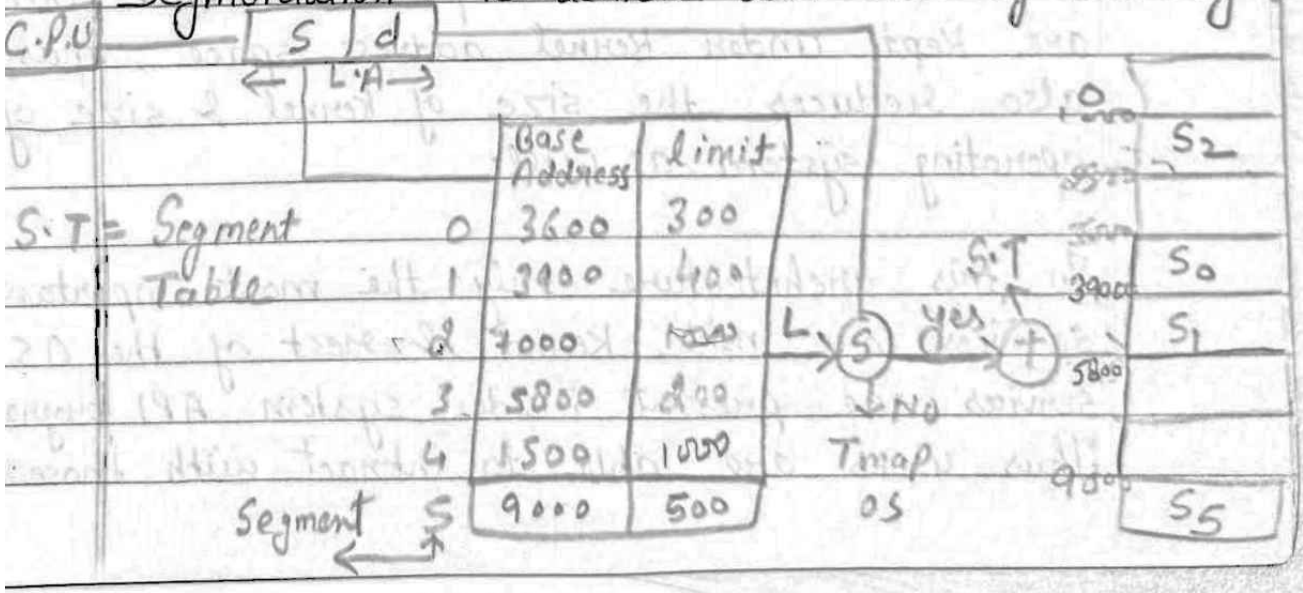
Searching :- Process ID & Page no. should match



Inverted page table
or frame table

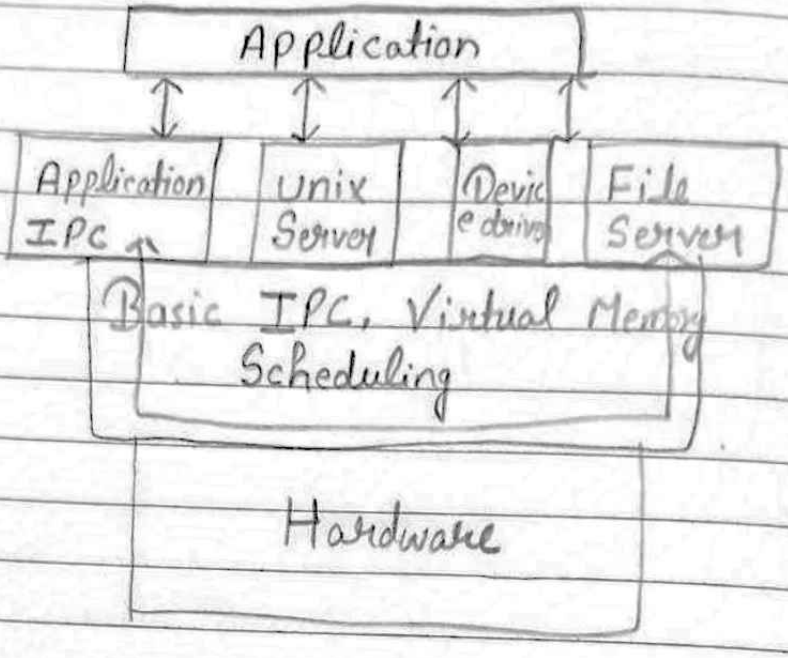
No. of Entry in Inverted page = No. of frame in P.A.S

Segmentation :- To achieve user view of Memory



L·A·S divide into various segments S & no. of bit required for no. of segments d & Segment size.

What is Microkernal



MicroKernal is one of the classification of the Kernel. Being a Kernel it manages all System resources. But in a microkernel, the user services & Kernel services are implemented in different address space. The user services are kept in user address space & Kernel services are kept under Kernel address space, thus also reduces the size of Kernel & size of operating system as well.

In this architecture only the most important services are inside Kernel & rest of the OS services are present inside system API program Thus users are able to interact with those

not so important Services within the system application. the microkernel is responsible for the most important services of operating System.

- I P C
- M. M
- C. P. U Scheduling