Date : / /
Page No.

O.S :- It is the most critical s/w used in a computer system.

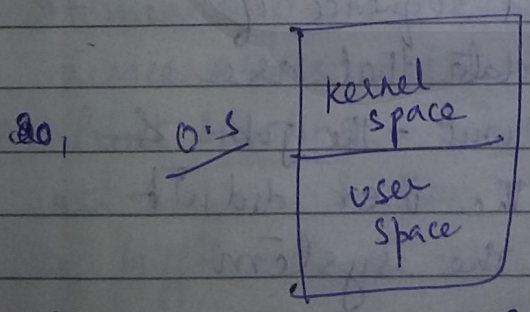
✓ O.S provides functions to assist in processing & execution of a program

✓ Kernel services, library services & application level services all are part of O.S.

① Processes run application-~~level services~~ which are linked with libraries that perform standard services.

② Kernel support processes by providing a path to the peripheral devices.
Kernel is the core of O.S & a permanent resident. It creates & terminates processes & responds to their request for services
Kernel manages input/output requests from s/w and translates them into data processing (meaningful info) inst^{n} for the CPU.

O.S consists of kernel space & user space (useful applications & utilities)

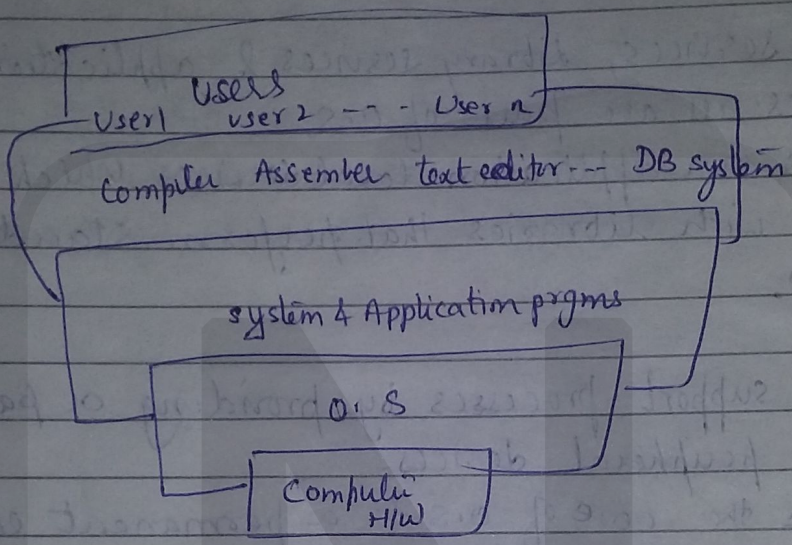


LINUX is kernel or O.S
[bcoz it doesn't include graphical desktops, text editor, compiler, file system utilities].

Date : / /
Page No.

O.S is intermediary b/w a user of a computer and the computer h/w.

Abstract view of components of O.S
(hide the lower level details)



OS hides the details of how the h/w operates thus making computer h/w relatively easy for an application programmer to use.

Types of O.S

1 Batch O.S :- Collection of Jobs is called a Batch. Job is predefined sequence of commands, programs & data that are combined into a single unit - the job - & submitted to the system. The user didn't interact directly with the system

means doing the same Job over & over again.

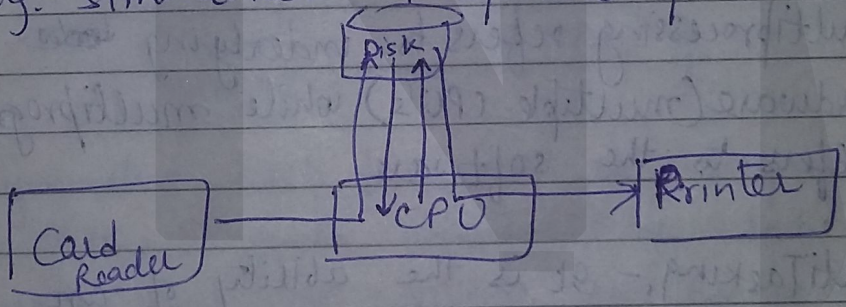
It is common on mainframe computer that was purchased specifically with massive repitive data processing in mind.

For eg A mainframe can be set to process 300 million persion statements.

Disad CPU is often idle.

(2) Time sharing Systems :- also called Multi Tasking. It uses CPU scheduling & multiprogramming. ~~to provide~~ In time sharing, each user is given a time-slice for executing his job in round-robin fashion. It allows context switch.

Spooling :- Simultaneous Peripheral op^n on line



It means putting jobs in a buffer. In this device access data at different rates.

eg Print Spooling :- In this, documents are loaded into buffer & then the printer pulls them off the buffer at its own rate.

Date : / /
Page No.

Multiprogramming :- There are one or more programs resident in computer's main memory ready to execute. Only one program at a time gets the CPU for execution while the others are waiting their turn. This is to utilize CPU time. The currently executing program

and transfer control to another program in line.

Main memory — partitions
↓
various jobs.

Multiprocessing :- It refers to CPU rather than running processes.

- Multiprocessing refers to underlying hardware (multiple CPU's) while multiprogramming refers to the software.

MultiTasking :- It is the ability of an operating system to execute more than one task simultaneously on a single processor time.

Date : / /
Page No.

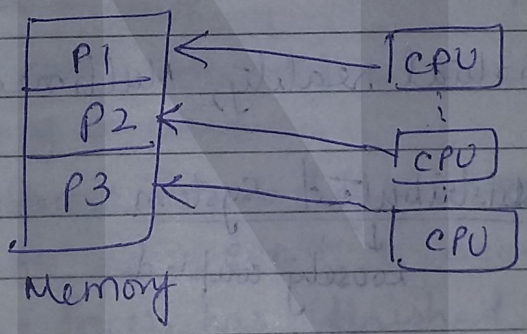
(M-S) or Parallel system or Tightly coupled system

Multiprocessor System :- It has more than one processor in close communication. They share the computer bus, system clock & i/p, o/p devices & sometimes memory.

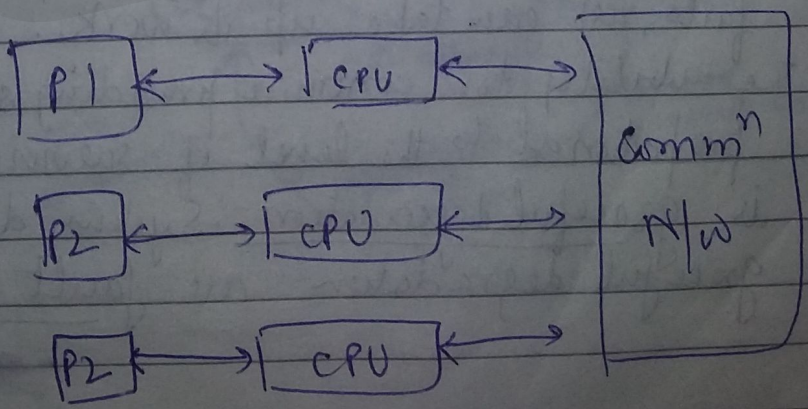
M.S are of 2 types

- Symmetric
- Asymmetric

Each processor runs an identical copy of the O.S. & they communicate with one another as needed. All CPU shared common memory.



In asymmetric, each processor is assigned a specific task. It uses master-slave relationship. A master processor controls the system and allocates work to slave processors.



(No Shared Memory)

Processor O.S Real Time

Date : / /
Page No.

Real Time System :- A RTOS is one that react to inputs and responds to them quickly. It has well defined, fixed time constraints.

2 types

1. Hard RTOS :- It guarantees that critical tasks to be completed on time. It requires that all delays in the system be bounded.

2. Soft RTOS :- It is a less restrictive type.

Eg Hard RTOS :- Flight simulation, Robotics.

Soft :- virtual reality, Multimedia

Parallel & distributed system
↓ ↓
Tightly coupled loosely coupled

Advantages of Multiprocessor System :-

1) Increased throughput :-
2) Increased Reliability :- If one processor fails, rest can take up its work.
The ability to continue providing services proportional to the level of surviving h/w is graceful degradation. System designed for graceful degradation are fault Tolerant.

Date : / /
Page No.

Process:- It is a program in execution.

A program is passive entity such as the contents of a file stored on a disk.

Process is an active entity, with a program counter specifying the next inst^{n} to execute and a set of associated resources such as CPU time, memory, files & I/O devices. These resources are allocated to a process when it is created or while it is executing.

Process state:- As a process executes, it changes state.

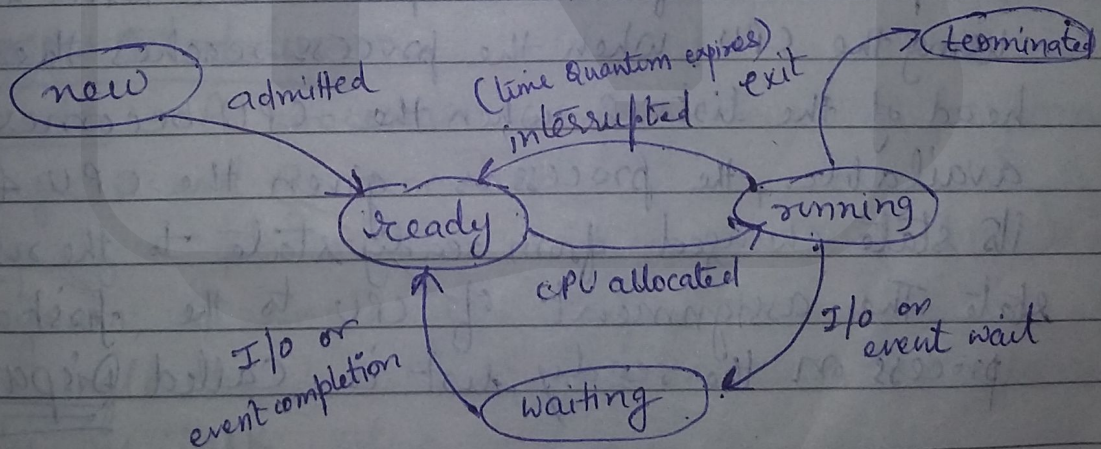
1) **New** :- Process is being created

2) **Ready**:- The process is waiting to be assigned to a processor.

3) **Running**:- CPU is allocated to process. Inst^{n} are being executed.

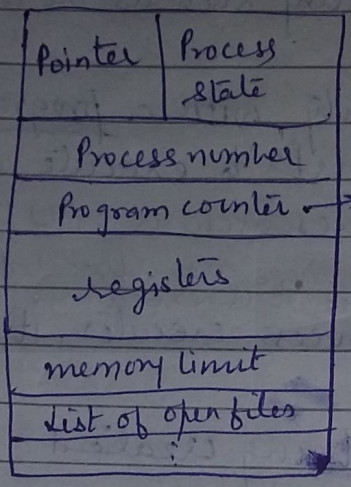
4) **Waiting**:- ~~Process are waiting for CPU~~. It is waiting for some event to occur.

5) **Terminated**:- Process has finished execution.



Date : / /
Page No.

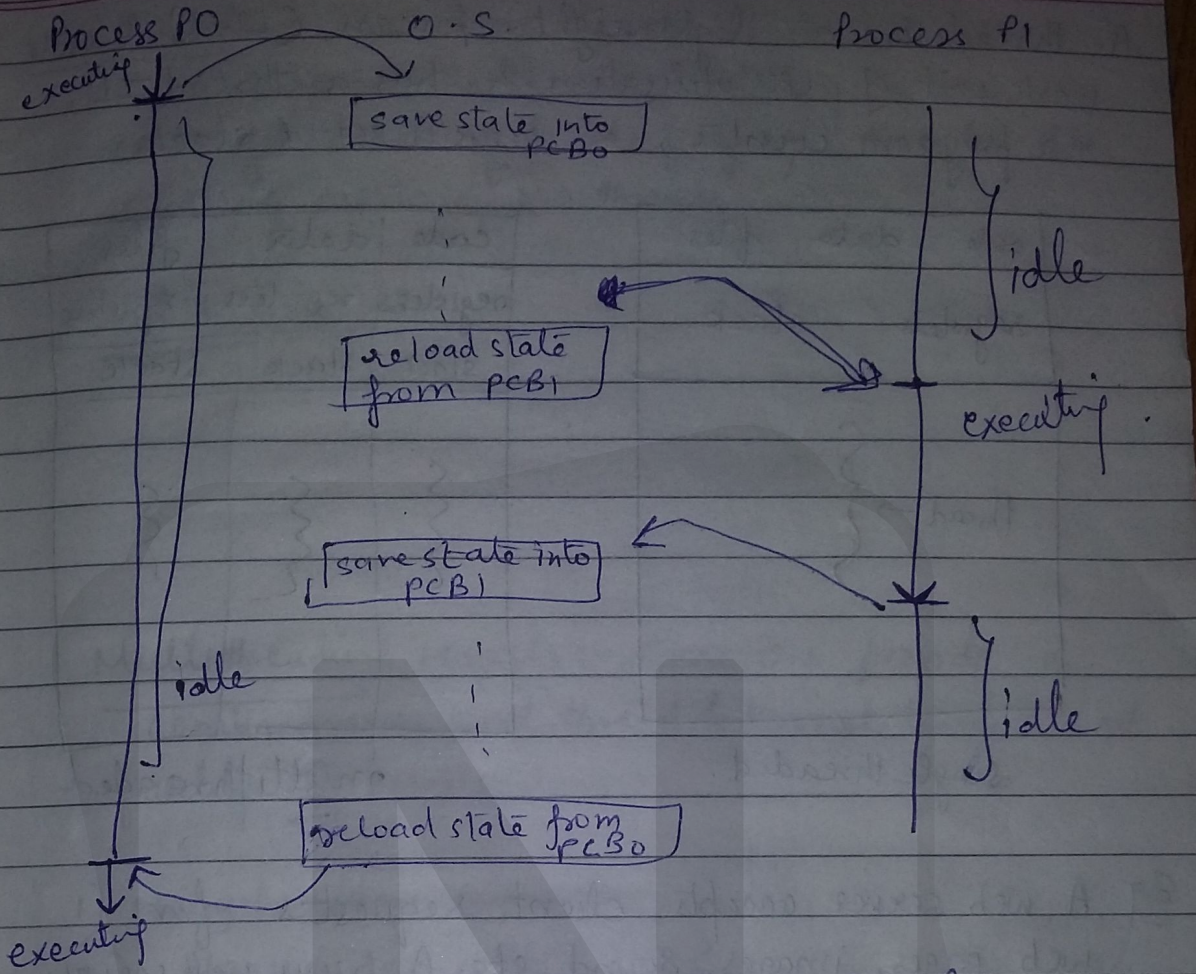
Process Control Block:- Each process is represented by in O.S by PCB.



PCB is just a repository of infor^n that may vary from process to process.

When a Job is admitted to the system, a process corresponds to the Job is created and inserted at the back of the ready list. The process gradually moves to the head of the ready list as the the processes before it complete their turns at using the CPU. When the process reaches the head of the list & when the CPU becomes available, the process is given the CPU & its state changed from ready state to the running state. The assignment of CPU to the first process on the ready list is called Dispatching & is performed by dispatcher.

Date : / /
Page No.



CPU switching from Process to Process.

~~Process scheduling:~~

Threads :- A process is a program that performs single thread of execution.

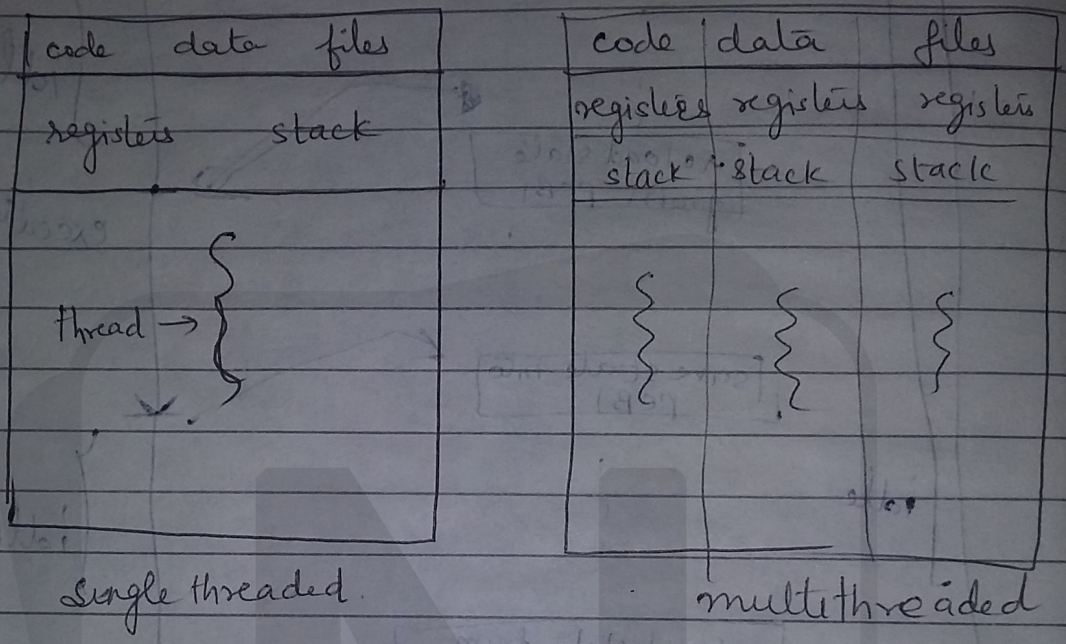
For eg If a process is running a word-processor program, a single thread of instr is being executed.

This single thread allows the process to perform only one task at a time.

That means, user couldn't simultaneously type characters & run the spell checker at the same time.

Date : / /
Page No.

A thread is a lightweight process (LWP) & is basic unit of CPU utilization & has a thread ID, a program counter, a register set & stack



Single threaded multithreaded

Eg 1 A web server accepts client requests for web pages, images, sound etc. A busy web server have several (million) of clients concurrently accessing it. If the web server ran as a single thread process, it would be able to service only one client at a time. Other clients have to wait enormously.

Eg 2 A web browser may have one thread display images or text while another thread retrieves data from the n/w.

2 Types of Threads :- User level.
Kernel level.

Date : / /
Page No.

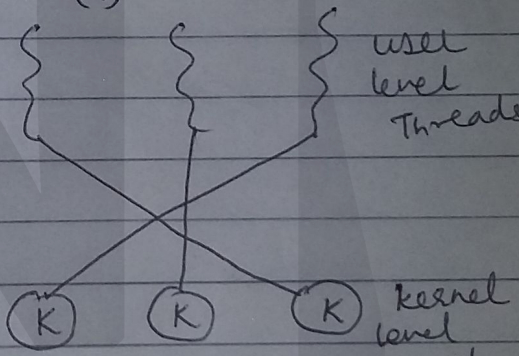
User Threads :- are above the kernel & without kernel support. The thread library contains code for creating & destroying threads, for passing message & data b/w threads.

Kernel Threads :-

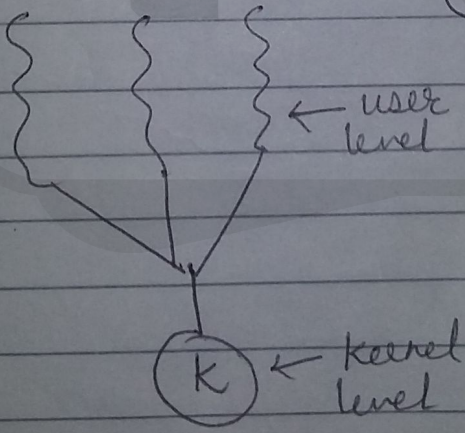
Multithreading Models :- Some O.s provide a combined user level thread & Kernel level thread faculty. Eg Solaris.

1) Many to Many
2) Many to one
3) one to one

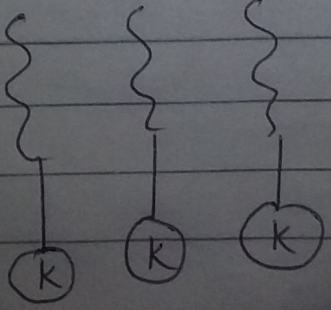
(1)



(2)



(3)



Date : / /
Page No.

Q

	List I	List 2
A	run → ready	1 not Possible
B	run → blocked	2 when a process terminates itself
C	blocked → run	3 when a process time Quantum expires
D	run → terminated	4 when a process issues an i/p o/p request

Codes A B C D

a) 1 4 3 2

b) 2 1 3 4

c) 3 4 1 2

d) 1 4 3 2

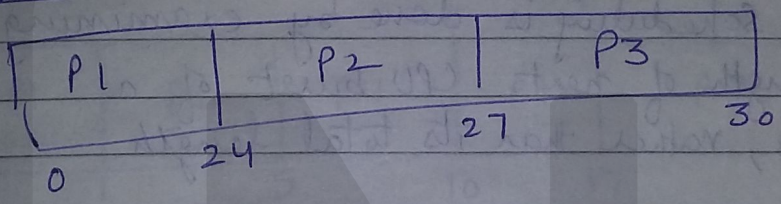
Date : / /
Page No.

Scheduling Algo

1) FCFS

Process	Burst Time
P1	24
P2	3
P3	3

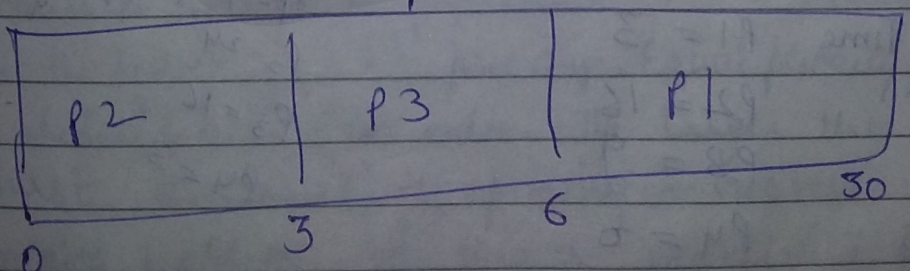
Gantt chart



Waiting Time P1 = 0
P2 = 24
P3 = 27

$AWT = \frac{0+24+27}{3} = \frac{51}{3} = 17\ ms$

Process	B.T.
P2	3
P3	3
P1	24



ATT
P1=30
P2=3
P3=6
$= \frac{39}{3} = 13\ ms$

Wt Time P1 = 6
P2 = 0
P3 = 3

$\therefore AWT = \frac{6+0+3}{3}$
$= 3\ ms.$

releases the CPU by terminating or by requesting I/O.

Date : / /
Page No.

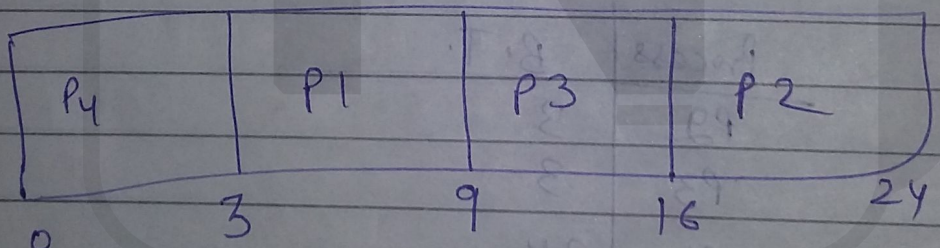
Convoy Effect :- ie all the other processes will wait for one big process to get off the CPU. This results in lower CPU utilization & device utilization that might be possible if the shorter processes were allowed to go first.

FCFS is nonpreemptive. Once the CPU has been allocated to a process, that process keeps the CPU until it

SJF :- Scheduling is done by examining the length of next CPU burst of a process, rather than its total length.

Process	B.T
P1	6
P2	8
P3	7
P4	3

If 2 processes have the same CPU burst then FCFS will break the tie.



Wt Time P1 = 3
P2 = 16
P3 = 9
P4 = 0

$= \frac{3+16+9+0}{4} = \frac{28}{4} = 7ms$

$P_1 = 9$
$P_2 = 24$
$P_3 = 16$
$P_4 = 3$

$\frac{52}{4} = 13ms$

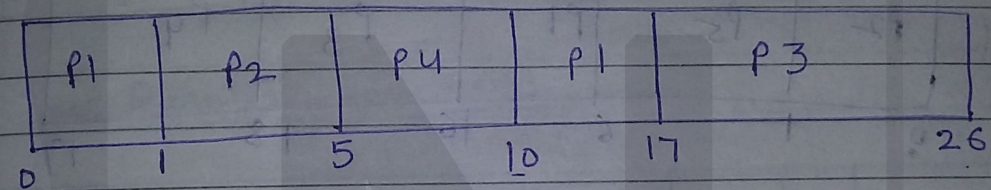
Difficulty with SJF is knowing the length of the next CPU request.

A preemptive SJF will preempt the currently executing process, whereas nonpreemptive SJF will allow the currently running process to finish its CPU burst.

Date
Page No.

Preemptive SJF (Shortest Remaing Time first)

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5



wt Time P1 = 10-1 = 9 ms
P2 = 1-1 = 0 ms
P3 = 17-2 = 15 ms
P4 = 5-3 = 2 ms

$P_1 = 17$
$P_2 = 4$
$P_3 = 26-2$
$P_4 = 10-3$
=

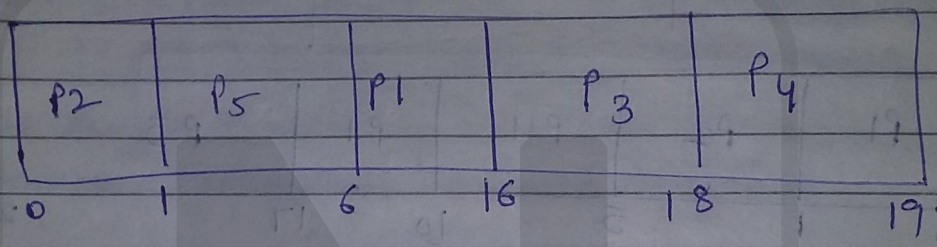
AWT = 9+0+15+2/4
= 26/4 = 6.5 ms.

Priority - In this, a priority is associated with each process & the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in FCFS order. The larger the CPU burst, the lower the priority & vice versa.
A major problem with this is indefinite blocking (starvation). A process that is ready to run but lacking CPU is considered blocked - waiting for the CPU. A Priority algo can leave some low

priority processes waiting indefinitely for the CPU.
A solution to this is aging in which you will gradually increase the priority of the process that wait in the system for

Date
Page No.

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2



Wt Time P1 = 6
P2 = 0
P3 = 16
P4 = 18
P5 = 1

$$AWT = \frac{6+0+16+18+1}{5} = 41/5$$

$$= 8.2 \text{ ms}$$

Ageing
Starvation

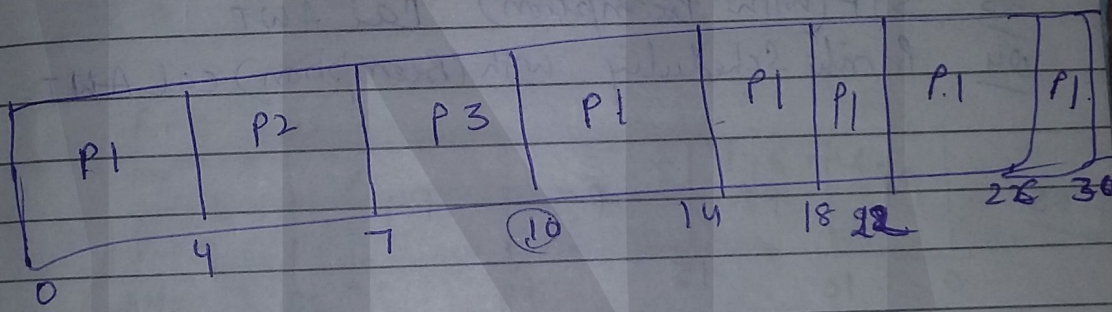
a long time. Eg If priorities range from 127 (low) to 0 (high), we would decrement the priority by one every 15 min.

Date: / /
Page No.

Round Robin : designed specifically for time sharing systems. In this, preemption is added to switch b/w processes. The ready queue is treated as circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of upto 1 time quantum. Here, we keep the ready queue as a FIFO queue.

Process	Burst Time of processes. New
P1	24 processes are added to
P2	3 the tail of ready queue.
P3	3

Time Quantum = 4 ms



Wt Time. P1 = 10 - 4 = 6

P2 = 4

P3 = 7

AWT = $\frac{6+4+7}{3} = \frac{17}{3} = 5.66$ ms.

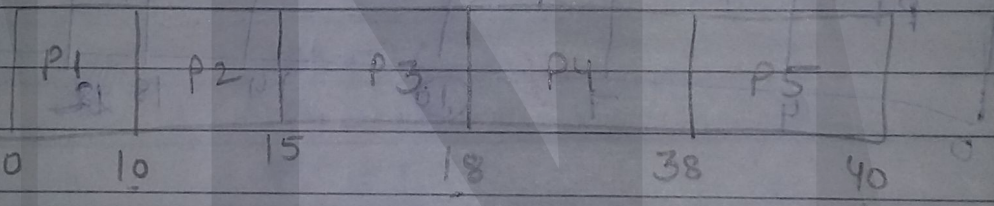
Date : / /
Page No.

Que Consider a set of 5 processes whose arrival time, CPU time needed & the priority are:-

Process	Arrival Time	CPU Time (ms)	Priority
P1	0	10	5
P2	0	5	2
P3	2	3	1 ← highest priority
P4	5	20	4
P5	10	2	3

Q1 Using FCFS, calculate AWT Ans 12.8 ms
Q2 Using SJF (without Preemption) cal. AWT 6.8
Q3 SJF (with Preemption) Cal AWT 5.6
Q4 Priority Scheduling with (Preemption) cal AWT.

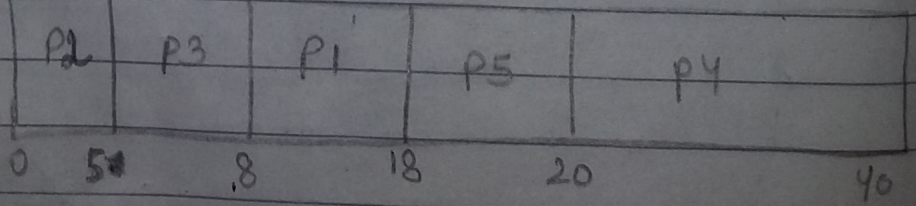
FCFS



P1 = 0
P2 = 10
P3 = 13
P4 = 13
P5 = 28

0+10+13+13+28 / 5 = 64/5 = 12.8 ms

SJF (Without Preem)

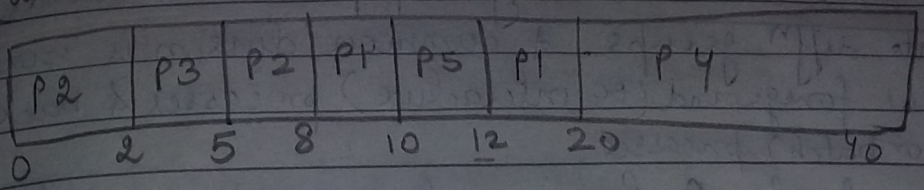


P1 = 8
P2 = 0
P3 = 3
P4 = 15
P5 = 8

8+0+3+15+8 / 5 = 34/5 = 6.8

Date : / /
Page No.

SJF (with Preemption)

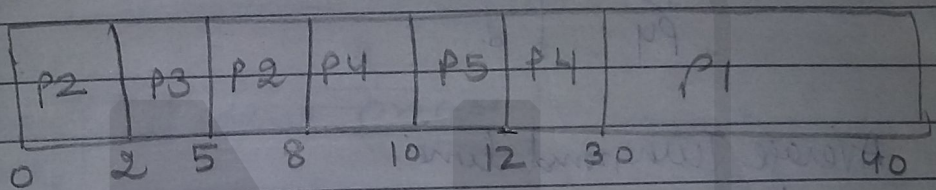


P1 = 12-2 = 10
P2 = 5-2 = 3
P3 = 2-2 = 0
P4 = 20-5 = 15
P5 = 10-10 = 0

$\frac{10+3+0+15+0}{5}$

$= \frac{28}{5} = 5.6$ ms.

Priority (Preemption)



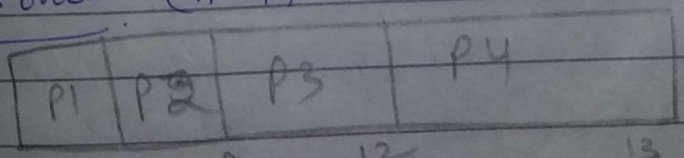
P1 = 30
P2 = 5-2 = 3
P3 = 0
P4 = 12-2-5 = 5
P5 = 0

$\frac{38}{5} = 7.6$ ms.

Ques

Process	CPU Burst Time	Arrival Time
P1	5	0
P2	3	1
P3	4	2
P4	1	3

FCFS order (AWT, ATT)



AWT

P1 = 0
P2 = 5-1 = 4
P3 = 8-2 = 6
P4 = 12-3 = 9

$\frac{19}{4} = 4.75$ ms.

Ans = 8 ms

ATT (when Job Submitted in the system)

P1 = 0
P2 = 5-1 = 3
P3 = 8-2 = 6
P4 = 12-3 = 9

$\frac{18}{4} = 4.5$ ms.

Date : / /
Page No.

(MQS)

Multilevel Queue Scheduling :- If processes are divided into diff^n groups. A common division is made b/w foreground (or interactive) processes & backgrd (or batch) processes.

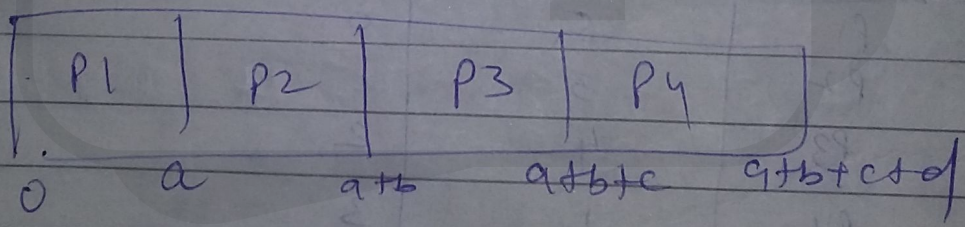
Ques P_1, P_2, P_3, P_4 consider $a < b < c < d$ (execution time)

	BT (ms)
P1	a
P2	b
P3	c
P4	d

Average Turnaround time

$$\frac{4a+3b+2c+d}{4}$$

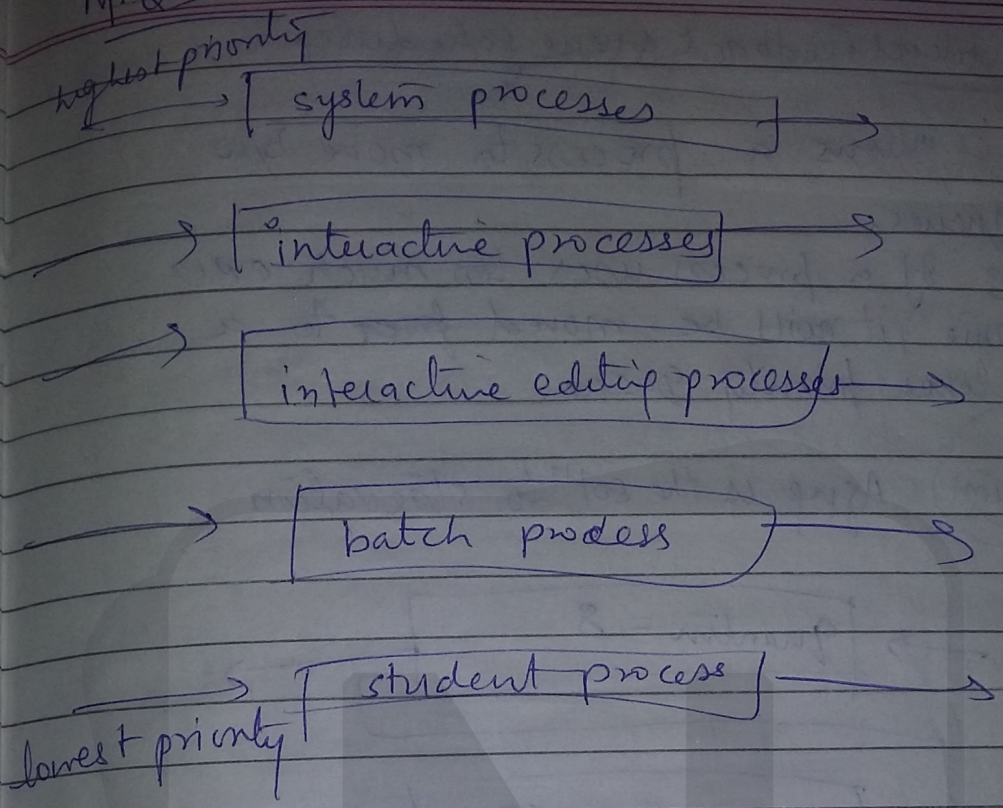
$$= \frac{(a-0)+(a+b-0)+(a+b+c-0)+(a+b+c+d-0)}{4}$$



Date : / /
Page No.

M.Q.S

highest priority



MQS

MQS partitions the ready queue into several separate queues. The processes are permanently assigned to one queue based on process priority, process type & memory size. Each queue has its own scheduling algo. for eg foreground queue might be scheduled by RR & background queue by FCFS.

Foreground queue gets the priority over background queue

Eg Solaris uses MQS

Date : / /
Page No.

Multilevel Feedback Queue scheduling

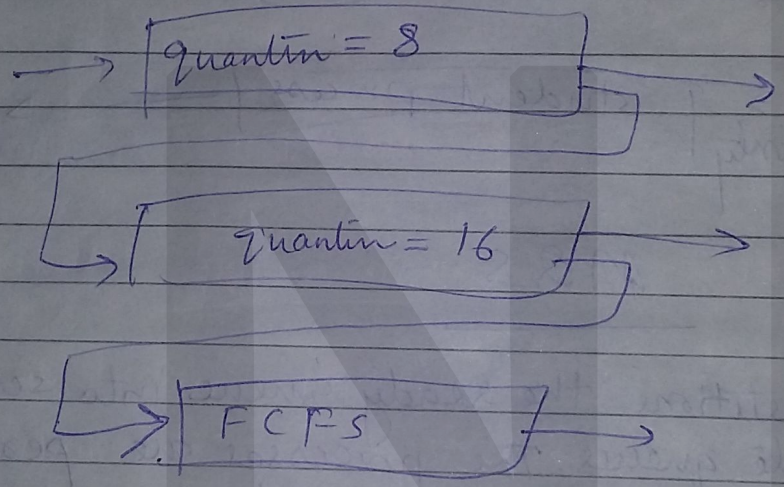
when

It allows a process to move b/w queues.

How If a process uses too much CPU time, it will be moved ~~from~~ to a lower priority queue.

Aging
Starvation

Aging is the sol^n to starvation.



MFQ

A process entering the ready queue is put in queue 0. A process in queue 0 is given a time quantum of 8 ms. If it doesn't finish within this time, it is moved to the tail of queue 1. If queue 0 is empty, the process at the head of queue 1 is given a time quantum of 16 ms. If it doesn't complete, it is preempted & put into queue 2. Processes in Q2 are run on FCFS basis, only

Date : / /
Page No.

When R0 & Q1 are ~~%~~ empty.

Date : / /
Page No.

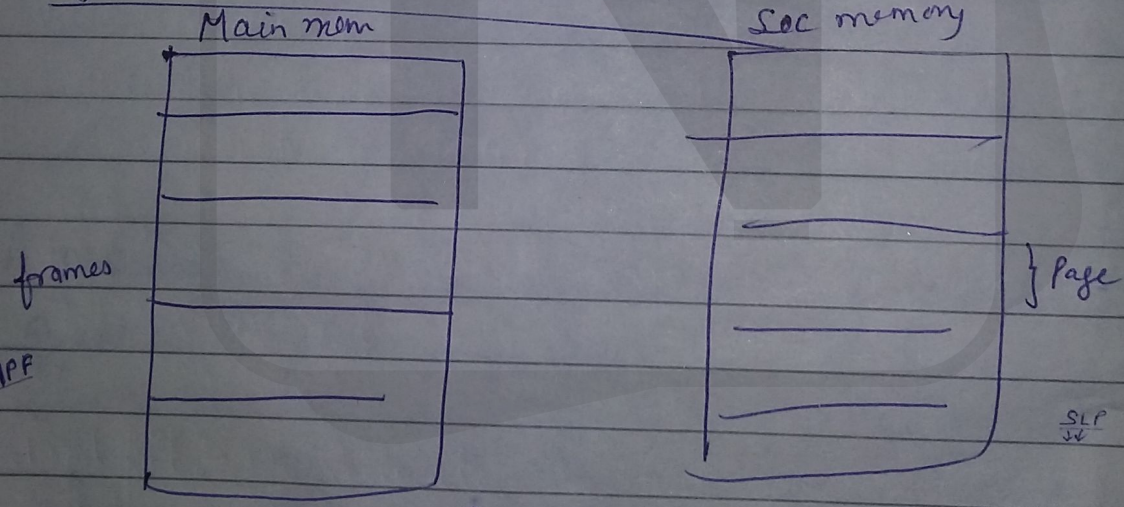
Memory Management

Through CPU scheduling, we showed that CPU can be shared by a set of processes. & we can improve CPU utilization & speed of computer's response to its users.

For this, we have to keep several processes in memory. Memory consists of large array of bytes each with its own address. CPU fetches instⁿ from memory according to the value of Program counter. Instⁿ is then decoded & may cause operands to be fetched from memory. After the instʳ has been executed on operands, result is stored back in memory.

The collection of processes that are waiting on the disk to be brought in memory for execution is called Input Queue.

logical vs Physical Address space



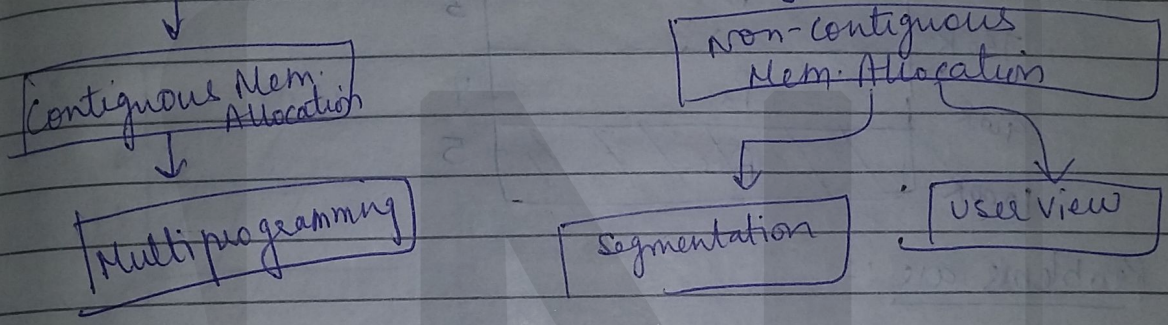
The page is brought from sec mem (disk) to main memory. This mapping from logical to physical is done.

Date : / /
Page No.

address generated by CPU is logical address (virtual address)
" " " Memory unit is Physical address.

The mapping from virtual address to physical address is done by a H/W device called MMU (Mem Mgmt Unit).

Mem. Mgmt. Schemes



Multiprogramming with fixed Partition

Processes reside in memory & various management policies are developed to provide the solution of this problem.

Partitioned Mem. Mgmt is one method in which physical memory is divided into various sectors called 'partitions' each of which may be allocated to a diff^n process.

(partitions are decided at the time of system System generation (at some time prior to the execution of user prgm), but once decided can't be changed

→ fixed partition

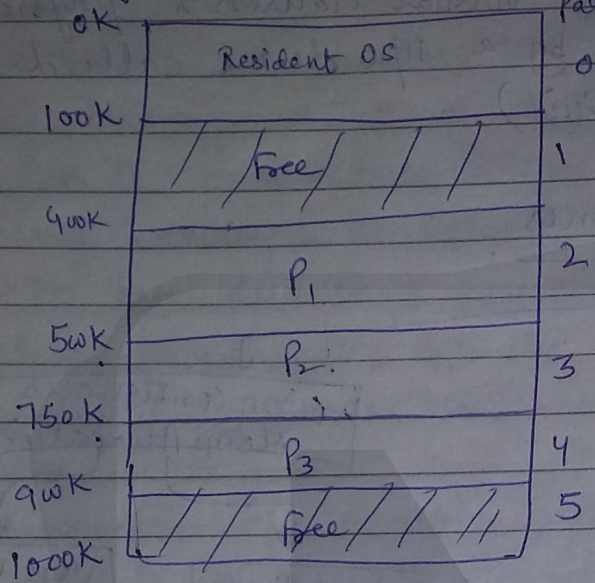
Date: / /

Page No.

Soln Ext frag gets involved in variable partitioning. Int-frag is stable

Partition decided on basis of available physical memory, desired degree of multiprogramming & size of processes.

eg



Problems are :-

1) How to allocate a specific partition for a given process?
2) What is done when no suitable partition is available for allocation?

For this, we can use the algo Best fit, Worst fit, & First fit

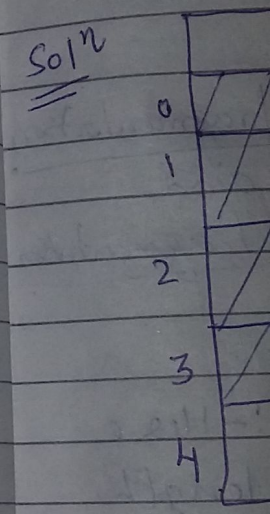
First fit :- We allocate the first hole that is big enough. Stop searching as you obtain a free hole that is large enough to accomodate the coming process.

Best fit :- Allocate the smallest hole that is large enough. Search the entire list

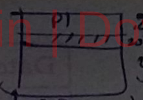
Worst fit:
produce

Ques For the
& 6
212 K,
& W

Soln



Best fit

In static, if  Process P1 comes so 20k is waste & is int frag. bcoz now this space can't be allocated to another process. Partitions are already done.

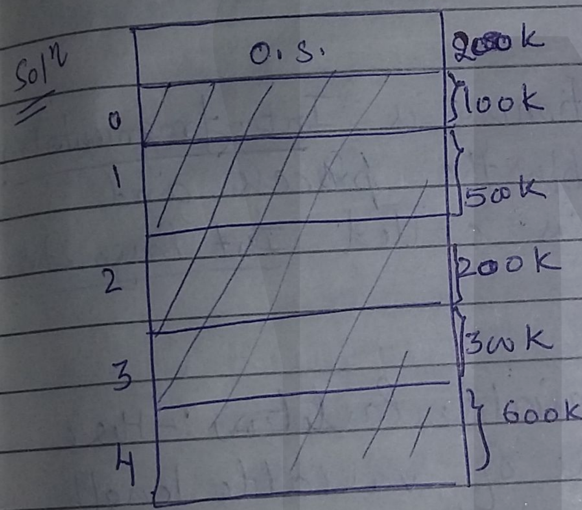
static.

unless it is sorted by size. This strategy produces the least size of memory space (fragments).

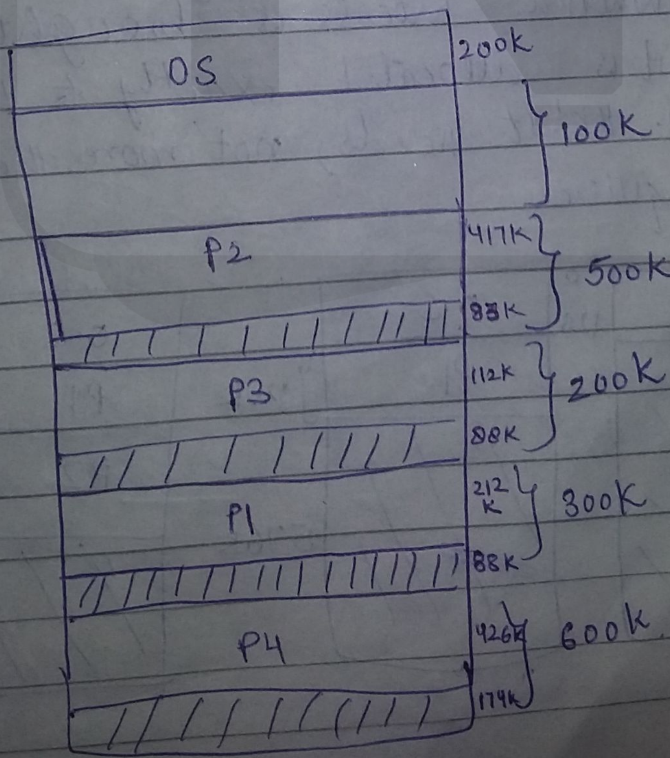
Worst fit :- We allocate the largest hole. It produces the largest size of unused mem. space.

Ques For the partition of 100k, 500k, 200k, 300k & 600k (in-order), place the processes of size 212k, 417k, 112k & 426k acc. to Best, First & Worst fit Algo.

Soln

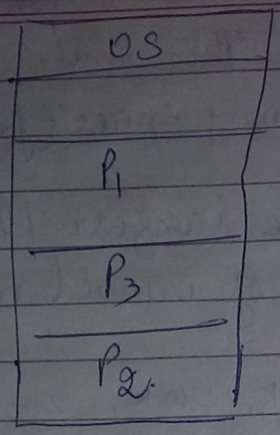


Best fit



Date : / /
Page No.

First fit

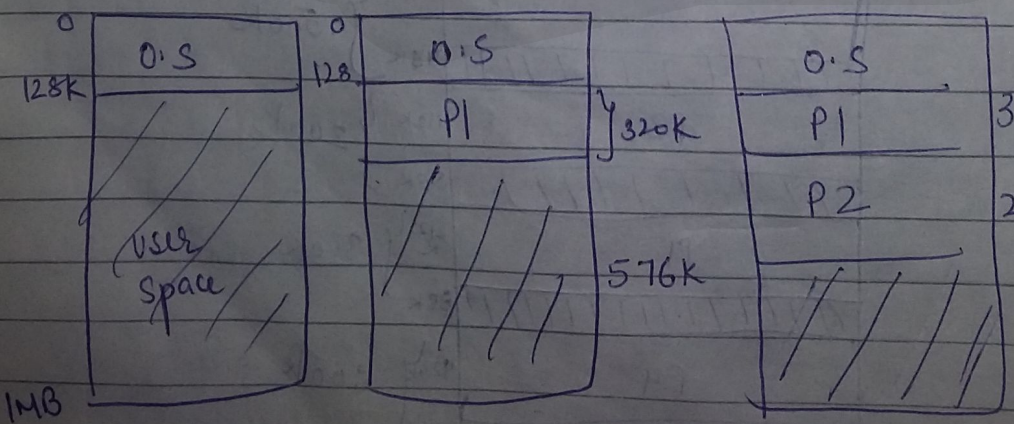


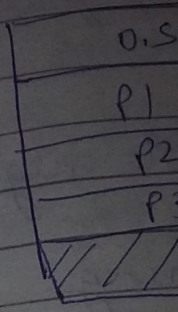
do it

Static partitioning is suitable for static env^r where workload is predictable. eg Banking system.

Problem with static partitioning is Int. fragmentation. The difference in size b/w the process and its allocated partition is called Int. fragmentation of memory.

Multiprogramming with variable Partition :- Here the partitions used are of variable length & no. . When a process is brought into mem., it is allocated exactly to that amt. of mem. that it needs, not more than that it requires.



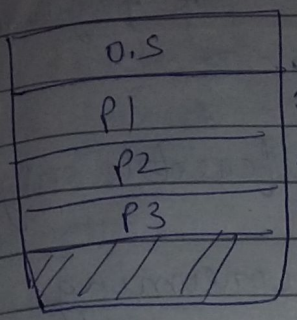


Now, P4
so, O.S
process
process wh
shortest
the de

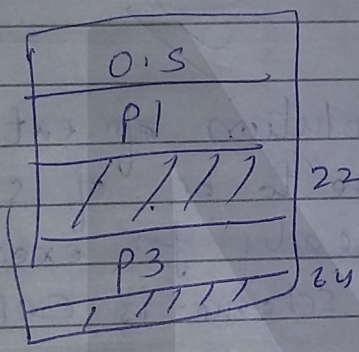
so, it

Now,
swa

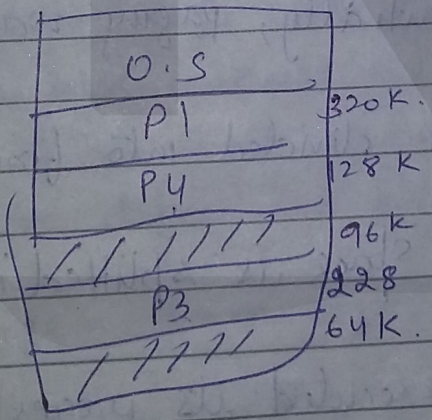
Date : / /
Page No.



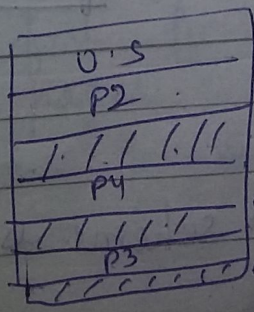
Now, P_4 comes but 64K is not enough for P_4(128K) so, O.S swaps out any of the three process whose execution is computed or the process which is not reqd presently acc. to short term scheduler. Let P_2 is swapped out. Now the diagram is



So, it is swapped (Process P_4)



Now, P_2 is again swapped in. So, Process P_1 is swapped out.



Date : / /
Page No.

Now, the mem. becomes more & more fragmented & is called ext. fragmentation.

① Solution is compaction :- [It is feasible only in dynamic env^{n}]. It is time consuming process.

② Soln is ← First fit / Best fit / First fit.

Soln for static partition mgmt - overlays or swapping is used.

Paging :- It is the solution for ext. fragmentation. If the chunks have to be of same size for all processes ready for execution then the mem. mgmt. scheme is called Paging.

Paging permits the physical address space to be non-contiguous. Initially, paging was used with the h/w support.
Physical mem is divided into fixed-size blocks called frames
Logical address space is splitted into pages.

When a process is executed its pages are loaded into any available mem. frame from the backing store.

An address translation scheme is used to map

the logical
Since, eac
page fra
need not
memory

CPU

Every
page n
an
conta
physic
page

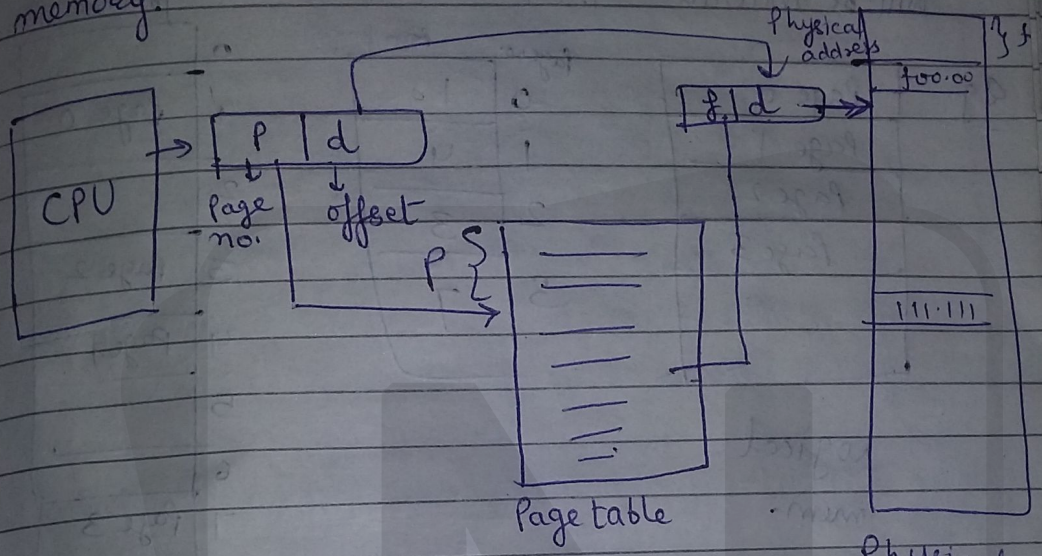
✓

✓ O·S

p	d
m-n	n

Date : / /
Page No.

the logical pages to their physical counterparts. Since, each page is mapped separately, diffn page frames allocated to a single process need not occupy contiguous areas of physical memory.



Paging h/w

Every address generated by CPU is divided into pageno(p) and offset (d). The pageno is used as an index into page table. The page table contains the base address of each page in physical mem. This page address is combined with page offset to define the physical mem. address

✓ Page size = frame size

✓ O.S. keeps track of all free frames. O.S needs n free frames to run a prgm of size n pages.

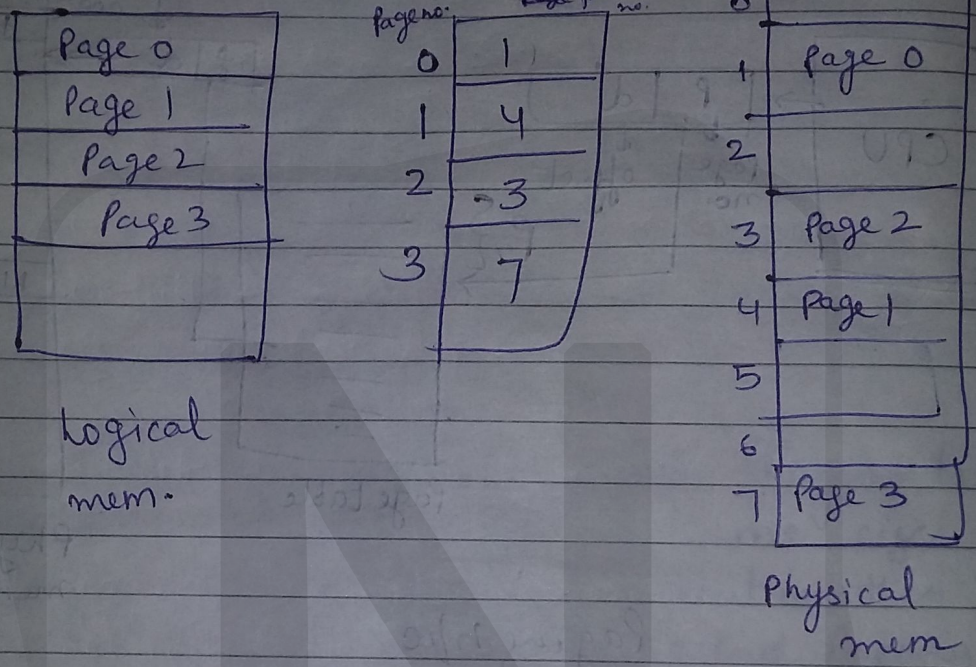
No. of pages = Logical Add. space / page size

Date : / /
Page No.

No. of frame = $\frac{\text{Physical address space}}{\text{frame size}}$.

✓ Whenever Paging is applied, the page table will be maintained. It contains the frame no.

Eg



Let Logical address 0 is page 0, offset 0.
Indexing into page table, we find that Page 0 is in frame 5.

✓ Logical address 0 maps to physical address 20

$20 = 5 \times 4 + 0$

5 → frame no. (frame 3); 4 → page size; 0 → offset

✓ Logical address 3 (Page 0, offset 3)

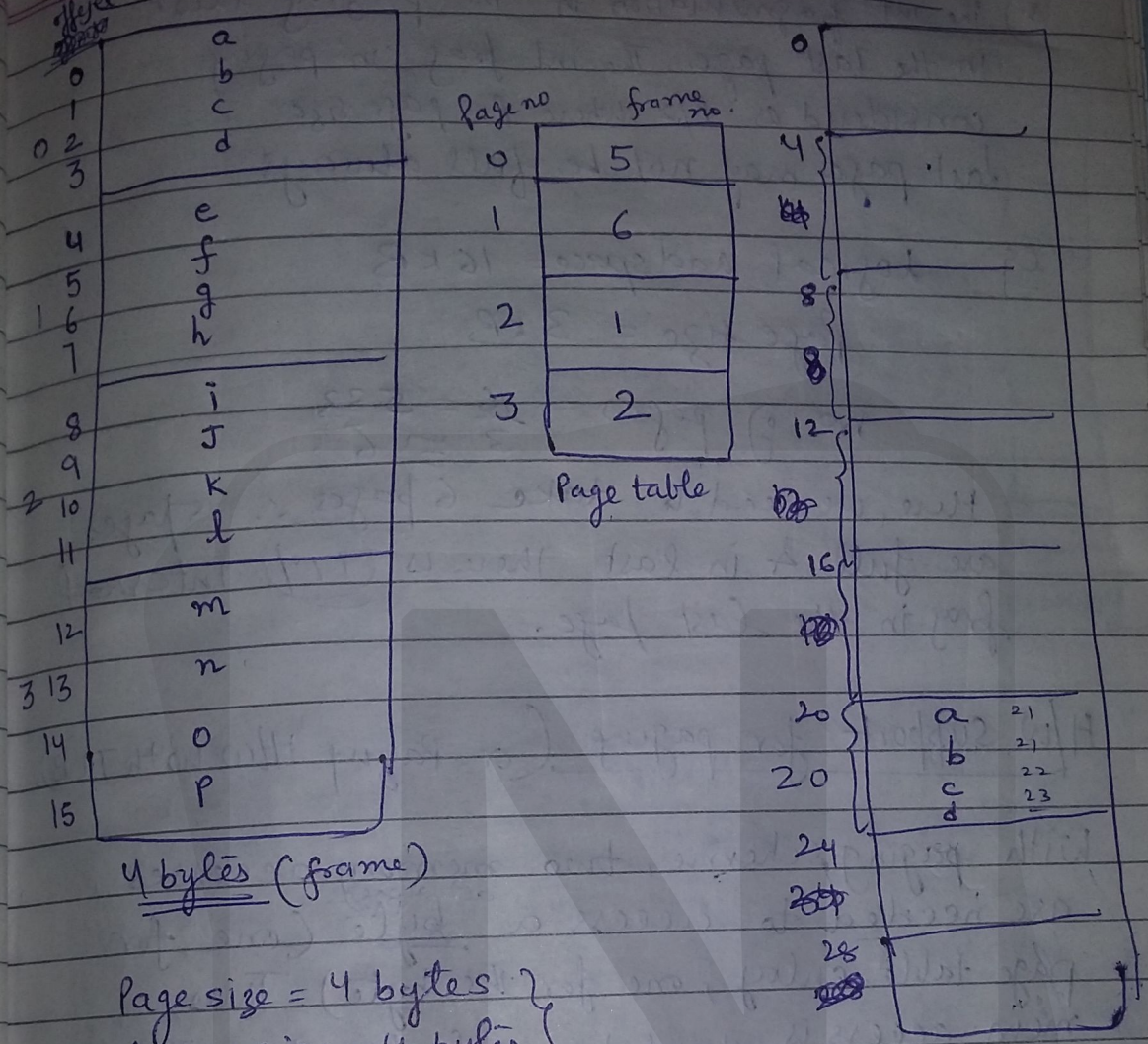
$= 5 \times 4 + 3$

$= 23$

frame 3 maps to page 23 or physical address 23.

Date : / /
Page No.

For Paging eg. for a 32 byte mem. with 4 byte page



Imp pts
1) Whenever the process is created the paging will be applied on process & the page table will be created & the Base address of Page Table will be stored in PCB.
2) Paging is wrt every process & every process will have its own page table.
3) page table of the process will be stored in the main memory.
4) No ext. frag. in paging.

Date : / /
Page No.

5) The int. fragmentation in the paging exist in the last page. The int. frag in paging considered as P/2 where P = page size. Last page may not be full always

eg Logical add space = 16 kB

Page size = 3 kB

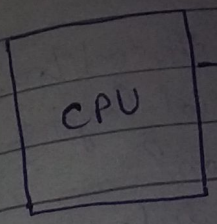
no. of pages = $\frac{16}{3} = 5.33 \simeq 6$

Here, we need to take 6 pages. ∴ 5 pages are full & in last there is (77%) internal frag in the last page.

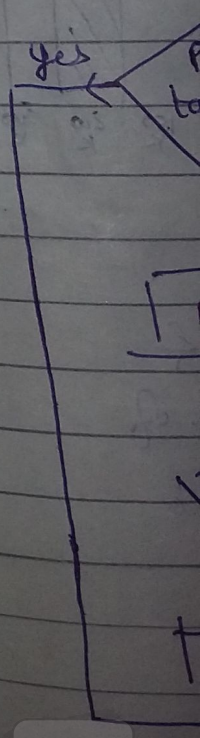
H/W Support for paging (or Paging H/W with TLB)

With paging scheme, two memory accesses are needed to access a actual page byte (one for page table entry, one for the byte). Thus, mem. access is slowed by a factor of 2. This delay would be intolerable under most cases.

So, TLB (Translation lookaside buffer) is used to improve the performance of paging. It is associative, high speed memory. When a logical address is generated by the CPU, its page no. is presented to the TLB. If the page no. is found, its frame no. is immediately available & is used to access memory. If Page no. is not in the TLB (TLB miss), a mem. reference to the page table must be made.

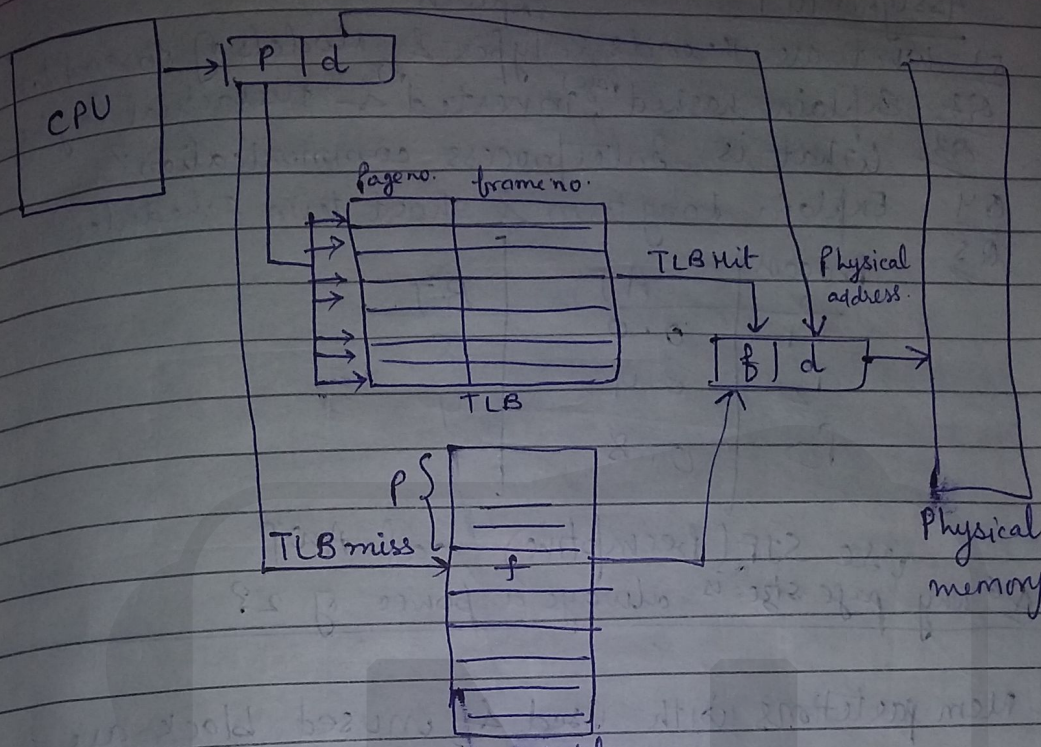


Flowchart

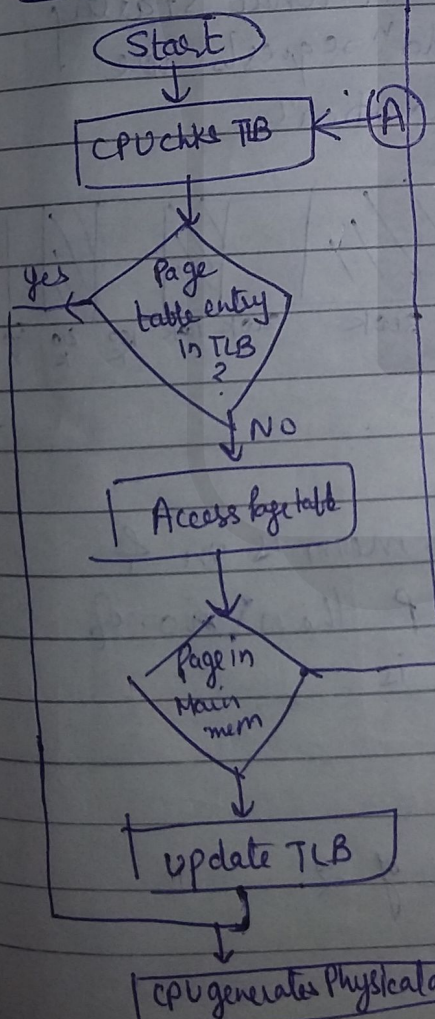


Paging H/W with TLB

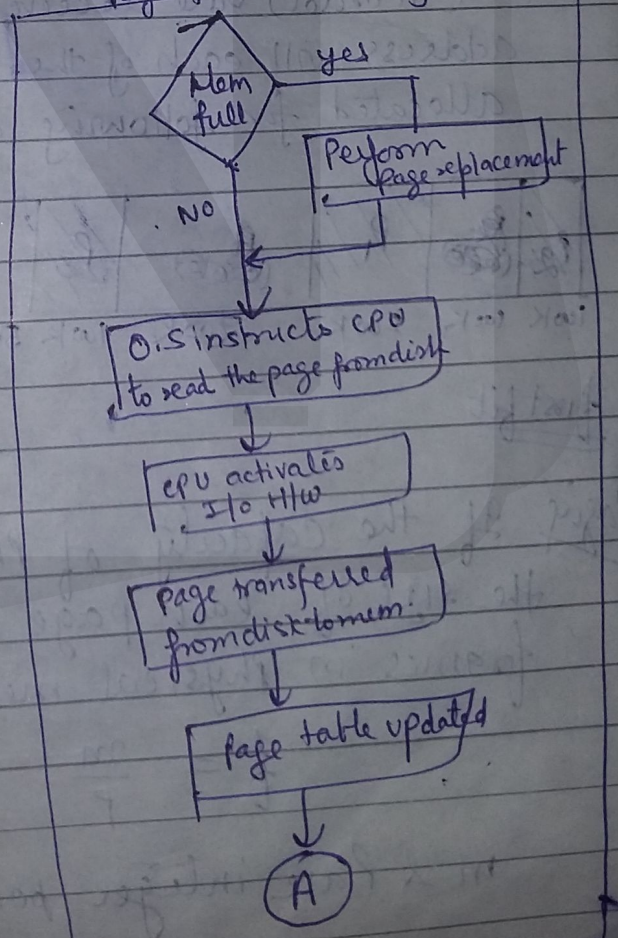
Date: / /
Page No.



Flowchart for TLB



Page fault Handling routine



Date : / /
Page No.

Assignment 1

Q1 What are threads ? Types & Models of thread too. explain

Q2 Explain hashed Page T, inverted PT & Hierachial Paging.

Q3 What is Interprocess communication?

Q4 Explain long term & short term scheduler.

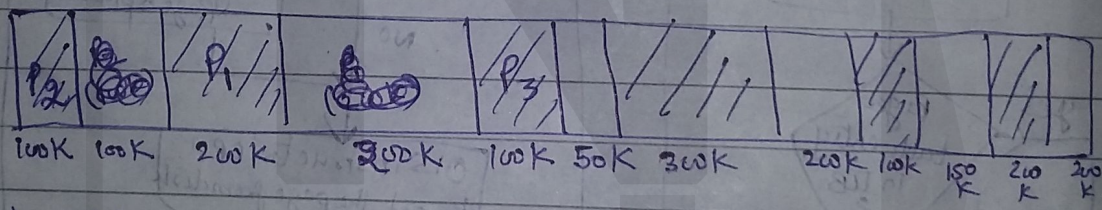
Q5

Process	AT	B.T
P_1	0.0	8
P_2	0.4	4
P_3	0.8	1

Use SJF (Preemptive), cal. AWT.

Q6 Why page size is always a power of 2?

Q Mem. partetions with used & unused block are given. Add^n requests for 200K, 100K & 50K (in order) are received. At what starting address will each of the add^n requests be allocated for following allocation?



First fit

Paging If the capacity of Phy. mem is m & the size of each page is p then no. of frames in physical mem is

$$f = \frac{m}{p}$$

m & p are integer power of 2.

Soln

Q Consi
of
(i)
(ii)
Q2 Ha
le

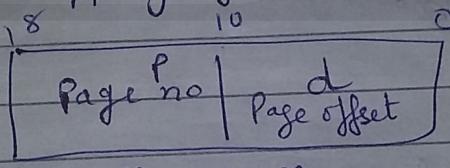
Date : / /
Page No.

Q1 Consider a logical address space of eight pages of 1024 words, each mapped onto a physical mem of 32 frames then

(i) How many bits are in logical address?
(ii) " " " " " Physical " .

Q2 How many pages will be reqd to convert a logical add of 18 bits into physical address during mapping if the size of each page is 1K.

Soln



Size of logical address is $2^{(m)}$ 18.

Page size is 2^n

Given logical address is 18 bits

∴ it is 2^{18}

$2^m = 2^{18}$

$m = 18$

Page size = 1K = 1024 = 2^{10}.

$2^n = 2^{10}$

$n = 10$ bits

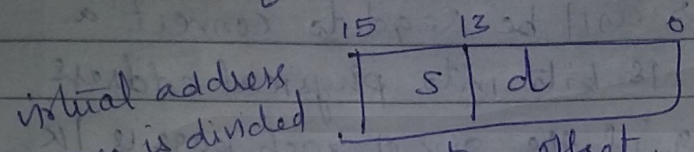
So, no. of pages reqd = 2^8.

$2^{18} - 2^{10} =$ 8 bits

Date : / /
Page No.

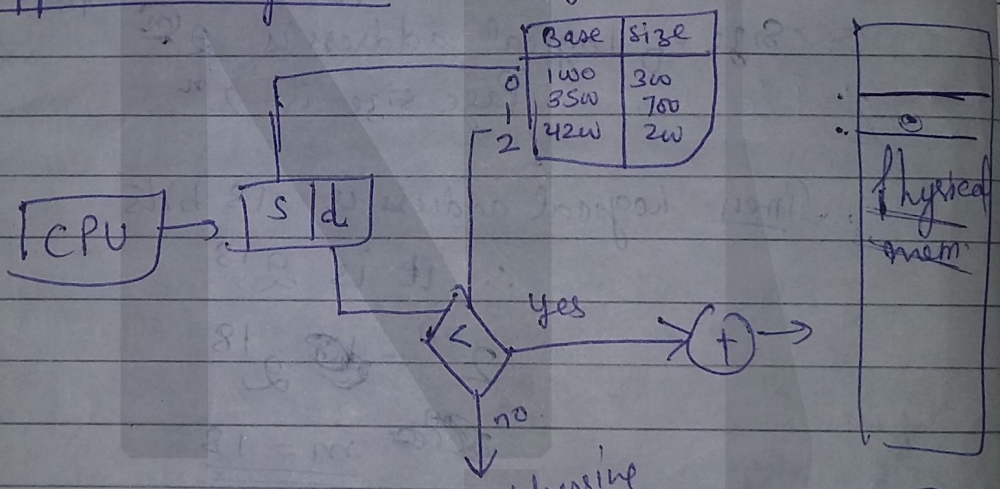
Segmentation = another technique

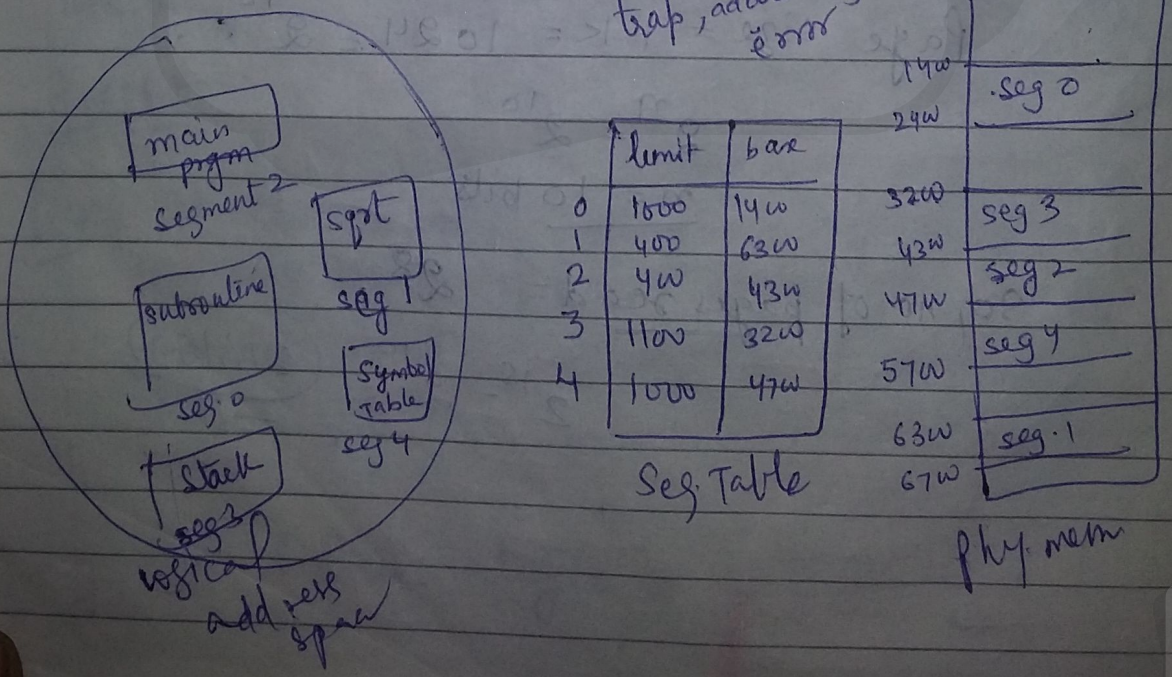
for non-contiguous mem. allocation. Segments are logical divisions of a progrem & hence are of variable size. In this, we divide the logical address space into segments. Division can be main program, set of subroutines, procedures, finctions & set of data structures.



virtual address space is divided into two or logical address space

H/W support in Segmentation





Date : / /
Page No.

Steps for Address Translation

1. Seg. no. is extracted from logical address by dividing the address into segment no. & offset. If there are 2^n segments, then n higher order bits in the logical address space specify the seg. no & the remaining no. give the offset.
2. If the seg. no. is greater than max^m no. of segments, a trap is generated to the O.S. It indicates that an attempt has been made to access an illegal smt.
3. Else, segment table entry is accessed:
   a) If segment offset is > (greater than) seg. size, a trap indicating "out of range" will be generated.
   b) If process doesn't have permission to access that segment, a trap indicating "protection violation" will be generated.
4. otherwise, Physical add. (segment base + offset) is returned.

Q Consider the segment table

Segment	Base	Length/limit
0	219	600
1	2300	14
2	900	100
3	1327	580
4	1952	96

Q Cal. Physical address for following logical address

0430, 110, 2500, 3400, 4112

Date : / /
Page No.

Soln For given logical address, first digit refers to segment no. & remaining digits refers to offset value (d)

i.e 0430
↓ ↓
segno offset value

So, Physical address for 0430 :
= base + offset (430)
= 219 + 430 = 649

for 110
↓ ↓
seg offset 2300 + 10 = 2310

for 2500
↓ ↓
seg offset 90 + 500 = 590 (illegal address).

340 = 1327 + 400 = 1727.

4112 illegal address (out of limit offset)

Virtual Memory

In contiguous & Non contiguous mem. allocation, the entire process was in the main memory at the time of its execution. Due to low priority, if a process is reqd. to be removed temporarily from main mem., then the entire process has to be swapped out. When it is again reqd. then the entire process is brought back into the main mem. called as swapped in. This Technique of temporarily removing inactive prgms. from main memory & bringing it again in main mem is swapping. This scheme is good

Date : / /
Page No.

but has a drawback. If the physical memory is limited then the no. of processes it can hold at any time & hence degree of multiprogramming becomes limited.

Thus, another method is evaluated in which we keep only a part of the process in the memory & other part on the disk (sec. storage). This Technique is virtual mem. mgmt.

So, Virtual mem. is an illusion that a compuler system possesses more mem. that it is actually having.

Virtual mem. is implemented by Demand Paging.

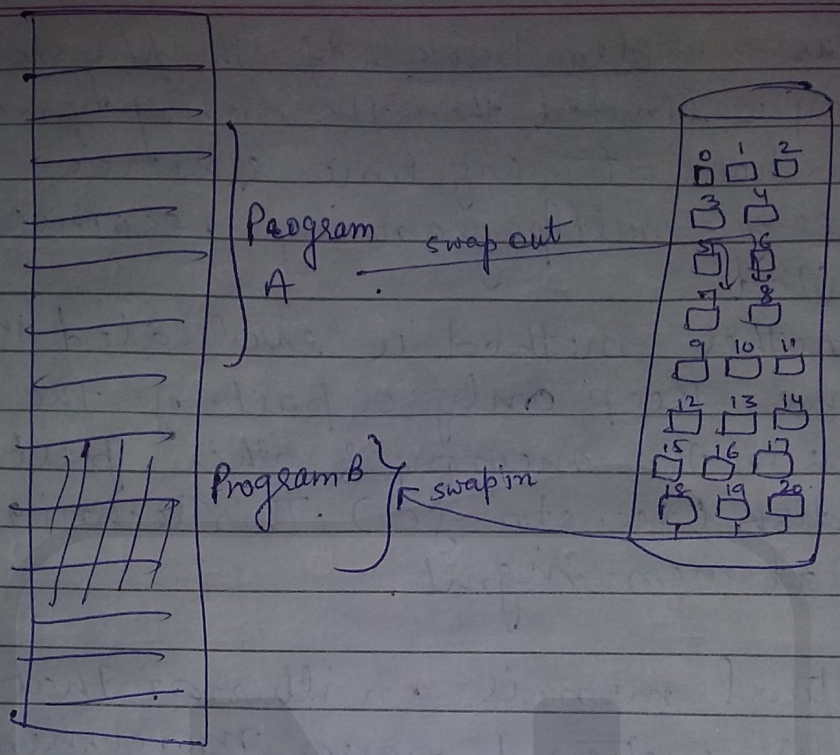
In Demand paging, a page is brought into memory for its execution only when it is demanded, else it remains on disk.

Req. is - that the complete program should be present in the backing storage (disk) in form of pages. A lazy swapper is used, which swap the pages when it is needed.

Adv

1) Reduced mem. requirement
2) Swap time is also reduced
3) Increased degree of multiprogramming.

Date : / /
Page No.



Transfer of a paged memory to contiguous disk space.

Disadv

Using this method, swapping is done using hit & trial method. for eg, if a program is having 5 pages: 0, 1, 2, 3, 4 & out of these only pages 0, 3, 4 are loaded into memory at the time of execution state. If the program tries to access the address of page number 2, then there will be an error as this page is not present in main memory since it was not loaded at the time of swapping from disk to main mem. This error is called page fault error.
In this case, operating system must bring the reqd page into the mem. before the execution of that instr can restart.

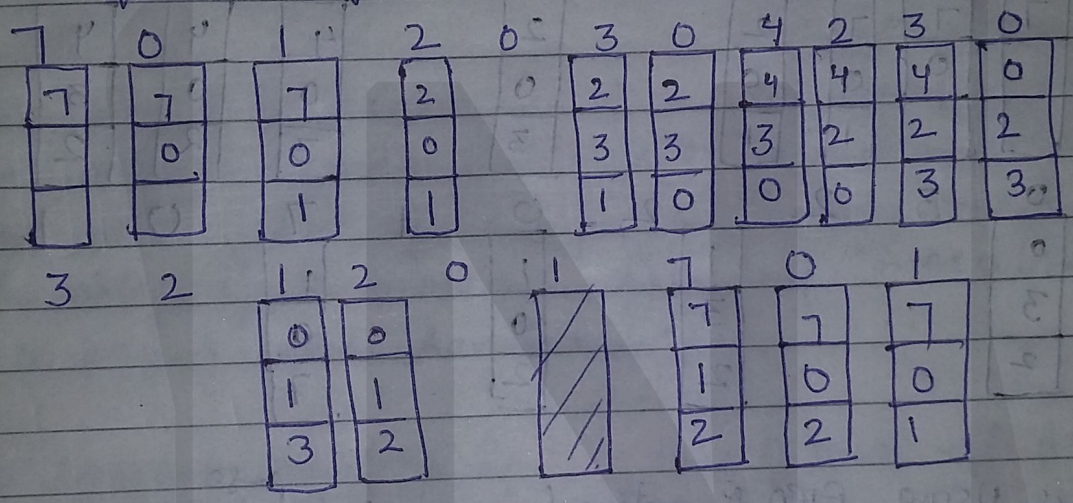
Date : / /
Page No.

Page Replacement Algorithm

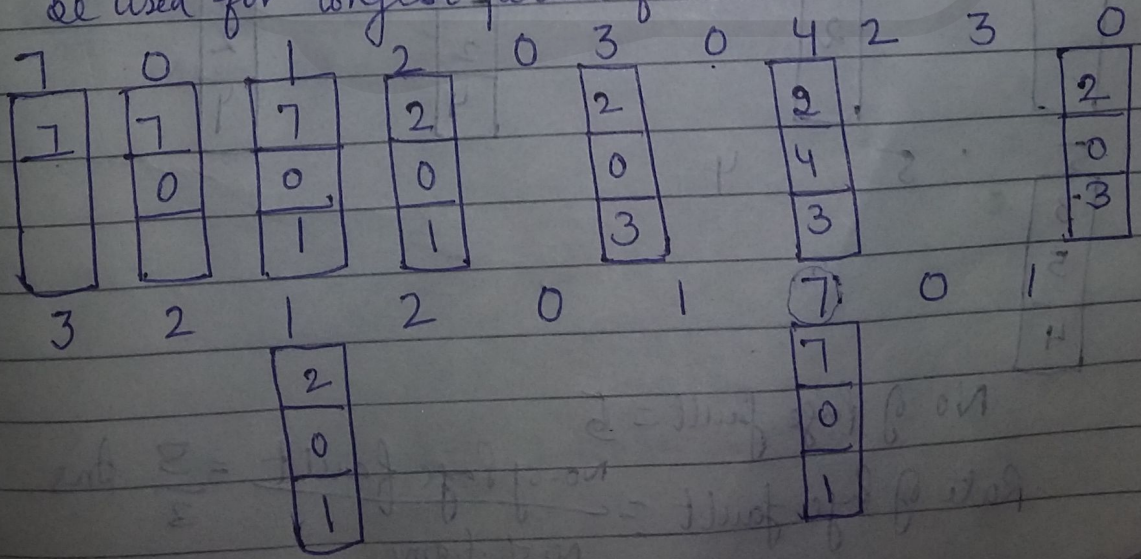
or Frame " "

① **FIFO** :- simplest, It associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. Page is inseted at tail of queue and being replaced at the head of queue.
Its performance is not always good.

Eg Reference String :-



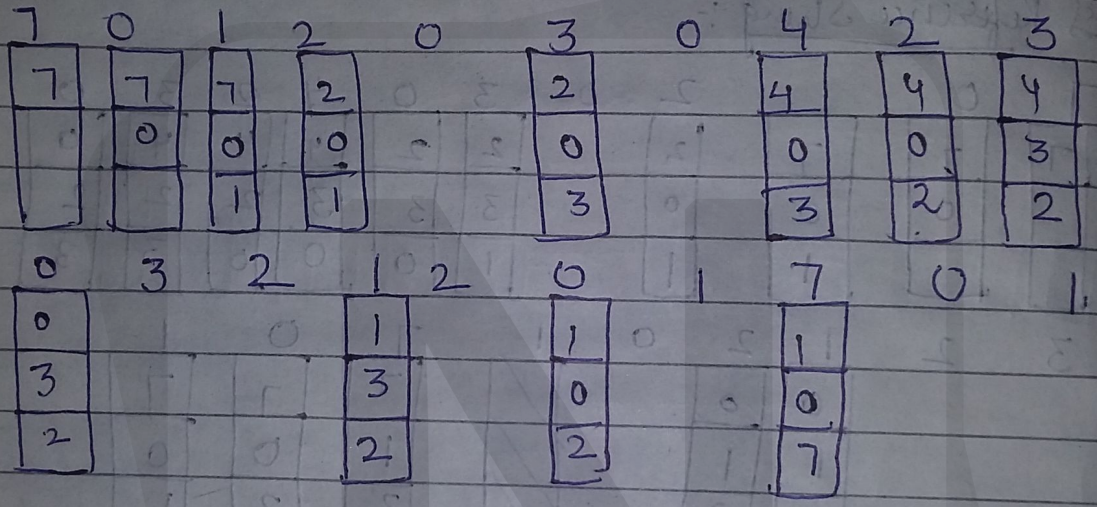
② **Optimal Page Replacement** :- It has the lowest page fault rate of all algorithms & never suffers from Belady's Anomaly. It says "Replace the page that will not be used for longest period of time".



But it is difficult to implement as it requires future knowledge of the reference string.

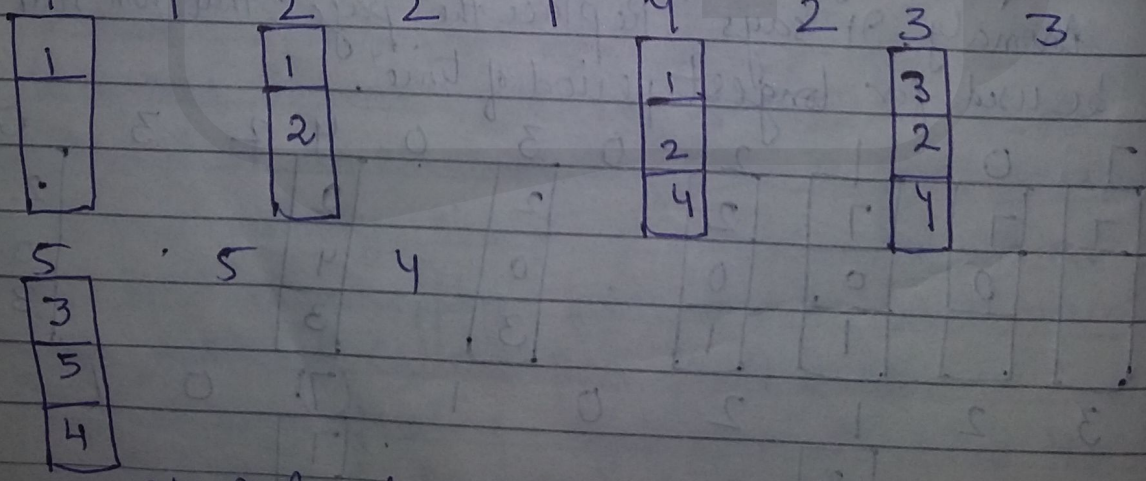
③ LRU Page Replacement (Least Recently used).

It associates with each page the time of that page's last use. When a page must be replaced, LRU chooses that page that has not been used for the longest pd. of time.



Ques Using FIFO & optimal, calculate page fault in each. All frames are initially empty & there are 2 frames & three frames.

Ref Str: No. of frame = 3.

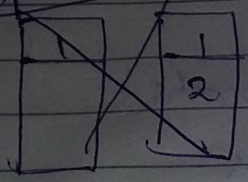


No. of Page fault = 5

Rate of Page fault = $\frac{\text{No. of Page fault}}{\text{No. of frame}} = \frac{5}{3}$ Ans.

Page No.

optimal



98
105

(2) frames Ans 6 Pagefault using FIFO
5 " " " Optimal.

Thrashing :- The phenomenon of moving pages from primary to secondary storage or vice-versa consumes a lot of computer's memory. This is thrashing if a process is spending more time in paging rather than executing.

Ques Consider the following sequence of mem. ref. from 460 words prgms 10, 11, 104, 170, 73, 309, 185, 245, 246, 434, 458, 364
Give the reference string assuming a page size of 100 words.

Soln Page size = 100 words.
Mem. ref. to will be made by Page

11	1
104	2
170	2
73	1
309	4
185	2
245	3
246	3
434	5
458	5
364	4

Date : / /
Page No.

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

(iii) Given size of Pr. Mem = 200 words
Page size = 100 words
No. of frames = $\frac{2w}{1w} = 2$ ✓

Now, use LRU & find no. of page faults.