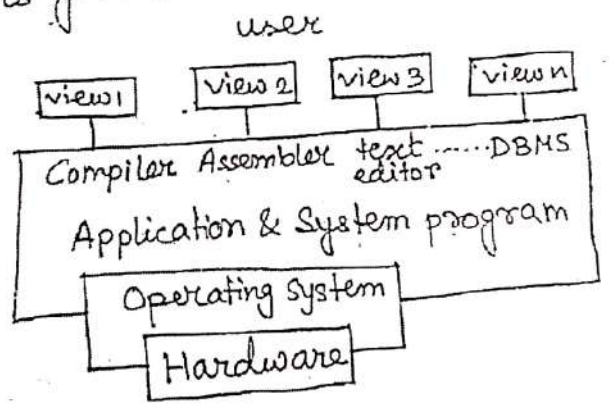
(1)

ADP – 12.7.11

⇒ What is operating system? What are the function of operating system?

Operating system is a program choice interface between hardware of the computer system and the user. Operating system provides a software platform on top of which other programs called application programs can run. The main goal of operating system is to execute user programs and make solving user problem easier.

Different components of computer system is given below:



Hardware – Physical component of computer system is called hardware. For eg: যন্ত্রপাতি CPU, memory, hard disk etc.

Operating system – Operating system is a program which controls and coordinate. different component of computer system. It provides an interface between the user and the hardware.

Application & System programs – Application program are those that are used to interact users. Such software are user specific. For eg: MS office, DBMS, Web Browser

System Programs – System programs are those that are mainly used for system related operations. For eg. compiler, assembler, loader, linker etc.

Users – People, computers, etc.

Functions of operating system are –

1) OS act as a interface between computer hardware and user. It is like a government that perform no useful task by itself. It simply provides an environment within which other programs can do useful task.

2) OS is a resource manager. It manages all the resources available ~~to~~ in the computer system.

3) OS is a resource allocator. It allocates all the resources required for running a program.

4) It controls and coordinates different I/O devices within the computer system.

5) OS perform scheduling of processes.

6) OS controls deadlock of processes.

7) It manages memory system.

8) It manages synchronization of processes

7) It provides directory structure of computer system.

8) It provides protection and security from unauthorized users.

9) What is process and program? Difference between process and programs. Draw the transition diagram of processes.

Ans:-

programs - Programs are set of instructions or code.

process - Process is a program in execution state. The program which is running called process.

Program	Process
i) Program is a set of instructions or code.	i) Process is a program in execution state.
ii) Program can be from any programming language.	ii) Process is always in machine language.

iii) Progra... static o...

iv) Progra... secondar...

v) Progra... passive

vi) The time... program...

Transit...

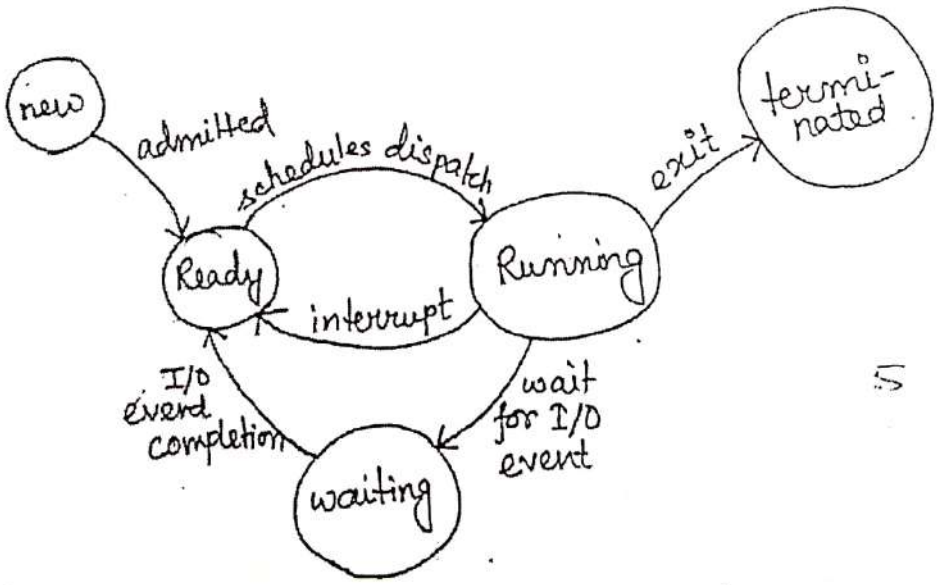
whe... states. ... transiti...

new

Program	Process
iii) Program is a static object.	iii) Process is a dynamic object.
iv) Program reside in secondary memory.	iv) Process reside in main memory.
v) Program is a passive entity.	v) Process is an active entity.
vi) The time span of program is unlimited.	vi) Time span of process is limited.

Transition diagram of process.

When a process executes it has many states. The following diagram shows the transition diagram of processes.



i) New - the process is being created.

ii) Ready - the process is waiting for execution. There is a ready queue that contains list of processes.

iii) Running - when the process is in the execution state. It is executed by the CPU.

iv) Waiting - The process is waiting for some event to occur.

v) Terminated - The process has finishes its execution.

15.7.11

3) Describe different scheduling algorithm.

There are many process scheduling algorithm. Some of them are given below :-

i) First come First serve (FCFS)
ii) Priority scheduling algorithm
iii) Shortest job first scheduling algorithm.
   a) Non preemptive SJF
   b) Preemptive SJF
iv) Round robin scheduling algorithm.
v) Multilevel queue scheduling algorithm.
vi) Multilevel feedback queue scheduling algorithm.

i) First Come First Serve

1) It is one of the simplest process scheduling algorithm.

2) The criteria of this algorithm is that the process that request the CPU first is hold the CPU first.

3) which process enter the ready queue first is serviced by the CPU first.

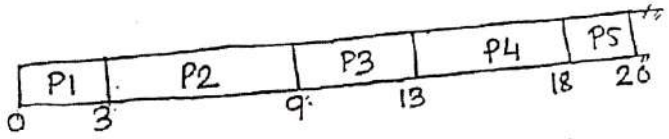
4) The implementation of FCFS is easily managed with a FIFO queue. When the CPU is free it is allocated to the process at the head of the queue. The running process is removed from the queue.

Example

Process	Burst time (ms)	Arrival time (ms)
P1	3	0
P2	6	2
P3	4	4
P4	5	6
P5	2	8

Calculate average turn around time and average waiting time.

ans:- Gantt chart for the FCFS. is given below.



Waiting time calculation

Waiting time for P1 = (0-0) = 0ms

Waitime time for P2 = (3-2) = 1ms

Waiting time for P3 = (9-4) = 5ms

Waiting time for P4 = (13-6) = 7ms

Waiting time for P5 = (18-8) = 10ms

Average waiting time = 23/5 = 4.6 ms

Turn around time calculation

Turn around time for P1 = (3-0) = 3

" " " P2 = (9-2) = 7

" " " P3 = (13-4) = 9

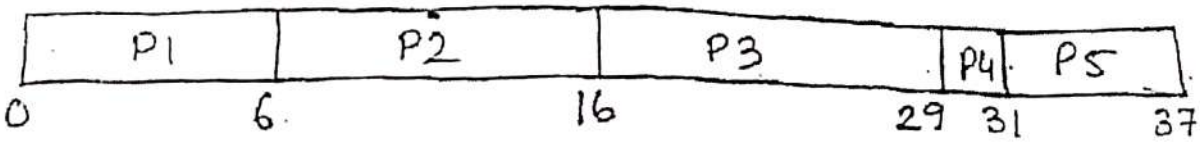
P4 = (18 − 6) = 12

P5 = (20 − 8) = 12

Average turn around time = 43/5 = 8.6

Process	Arrival Time (ms)	Burst time (ms)
P1	0	6
P2	5	10
P3	7	13
P4	11	2
P5	13	6

Gantt chart for the FCFS is given below.



Waiting time calculation

Waiting Time for P1 = (0 − 0) = 0 ms

" " " P2 = (6 − 5) = 1 ms

" " " P3 = (16 − 7) = 9 ms

" " " P4 = (29-11) = 18 ms

" " " P5 = (31-13) = 18 ms

Average waiting time = 46/5 = 9.2 ms

Turn around time calculation.

Turn around time for P1 = (6-0) = 6 ms

" " " P2 = (16-5) = 11 ms

" " " P3 = (29-7) = 22 ms

" " " P4 = (31-11) = 20 ms

" " " P5 = (37-13) = 24 ms

Average turn around time = 83/5 = 16.6 ms.

ii) Priority scheduling algorithm.

In priority scheduling algorithm a priority number is associated with each process. Depending on the priority number CPU allocate those process.

Equal priority ~~number~~ processes are scheduled in FCFS order.

Priority number are generally

set by two ways.

a) External priority
b) Internal priority

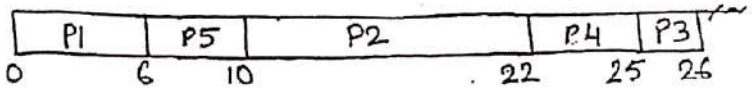
a) External priorities are given by the external users. It is generally 0 to 4095.

b) Internal priority number are set by the system.

Lowest priority number is used to specify highest priority for the schedule.

Process	Burst Time	Priority Number
P1	6	2
P2	12	4
P3	1	5
P4	3	4
P5	4	3

Calculate average waiting time and average turn around time.



Ex

Waiting time for P1 = 0

" " " P5 = 6

" " " P2 = 10

" " " P4 = 22

" " " P3 = 25

Average waiting time = 63/5 = 12.6

Turn around time for P1 = 6

" " " " P5 = 10

" " " " P2 = 22

" " " " P4 = 25

" " " " P3 = 26

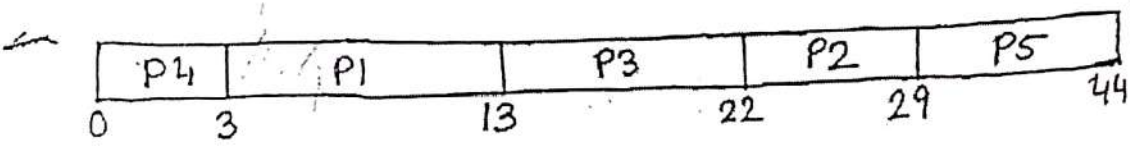
∴ Average turn around time = 89/5

= 17.8 ms

13

Process	Arrival Time	Burst time	Priority
P1	3	10	2
P2	2	7	3
P3	1	9	2
P4	0	3	1
P5	2	15	4

Gantt chart



Waiting time for P4 = (0-0) = 0

P1 = (3-3) = 0

P3 = (13-1) = 12

P2 = (22-2) = 20

P5 = (29-2) = 27

Average waiting time = 59/5 = 11.8

14

Turn around time for P4 = (3-0) = 3

" " " " P1 = (13-3) = 10

" " " " P3 = (22-1) = 21

" " " " P2 = (29-2) = 27

" " " " P5 = (44-2) = 42

Average turn around time = 103/5 = 20.6

ADP

44

15

Shortest job first scheduling algorithm

A different approach of CPU scheduling is shortest job first scheduling algorithm.

i) The criteria of this algorithm is which processes have smallest CPU burst time is assigned to that process.

ii) This algorithm associates with each process. The length of the next CPU burst time.

There are two types of shortest job first algo –

a) Non preemptive
b) Preemptive

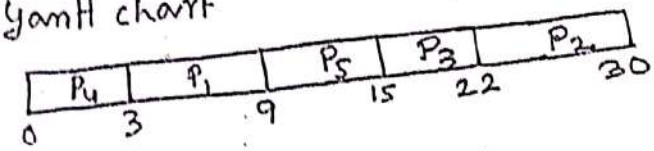
a) Non Preemptive – In case of non preemptive SJF no preemption is allowed between process. When a process enter the execution state it will not terminate until it finishes its execution.

b) Preemptive SJF – In case of preemptive SJF preemption is allowed between processes. When a process enter the execution state it can be preempted to make room for another process.

Example of non preemptive SJF

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3
P_5	6

Gantt chart



Waiting time calculation –

Waiting time for $P_4 = 0$

" " " $P_1 = 3$

$P_5 = 9$

$P_3 = 15$

$P_2 = 22$

Average waiting time $= \frac{49}{5} = 9.8$

Turnaround time calculation

Turn around time for $P_4 = 3$

" " " $P_1 = 9$

$P_5 = 15$

$P_3 = 22$

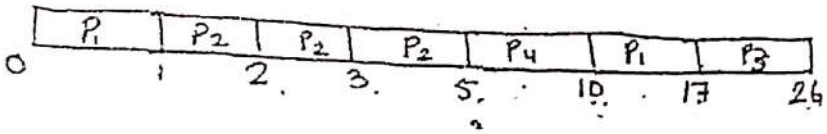
$P_2 = 30$

Average turn around time =

Example preemtive SJF

Process	Arrival time (ms)	Burst time (ms)
P_1	0	8 → 7
P_2	1	4 → 1
P_3	2	9
P_4	3	5

Gantt chart



Waiting time for $P_1 = (10-1-0) = 9$ ms

" " " $P_2 = (3-2-1) = 0$ ms

" " " $P_3 = (17-0-2) = 15$ ms

" " " $P_4 = (5-0-3) = 2$ ms

Average waiting time $= \frac{26}{5} = 6.5$

Turn around time calculation

Turn around time for $P_1 = (17-0) = 17$

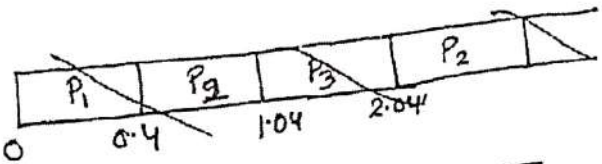
" " " " $P_2 = (5-1) = 4$

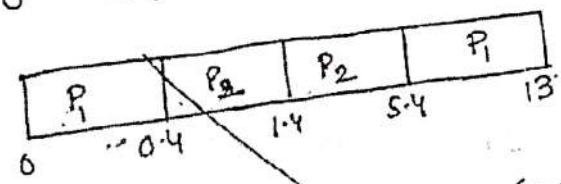
$P_3 = (26-2) = 24$

$P_4 = (10-3) = 7$

Average turn around time = $\frac{52}{4}$ = 13.

Process	Arrival time	Burst time
P_1	0	8 (7.6)
P_2	0.4	4 (3.4)
P_3	1	1

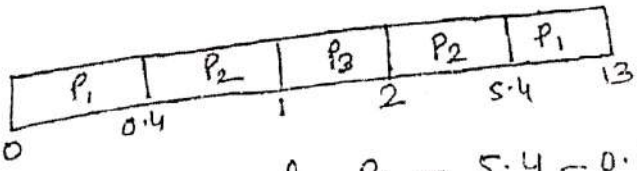




Waiting time for P_1 = (5.4 − 0.4 − 0) = 5

" " P_2 = (1.4 − 0 − 0.4) = 1

" " P_3 = (0.4 − −1) =



waiting time for P_1 = 5.4 − 0.4 − 0 = 5

P_2 = 2 − 0.6 − 0.4 = 1

P_3 = 1 − 0 − 1 = 0

Average waiting time = 6/3 = 2

Turn Around Time for $P_1 = (13-0) = 13.0$ ms.

$P_2 = (5.4-0.4) = 5.0$ ms.

$P_3 = (2-1) = 1$ ms.

$19.0/3$ ms.

$= 6.33$ ms.

Round Robin scheduling algorithm

i) The Round Robin scheduling algorithm is designed specially for time sharing system.

ii) It is similar to FCFS (First come First Serve) but preemption is added to switch between processes.

iii) A small unit of time called quantum time (slice time) is defined.

iv) A time quantum is generally 10 to 100 ms.

v) The ready queue is considered as circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for the time interval upto 1 time quantum.

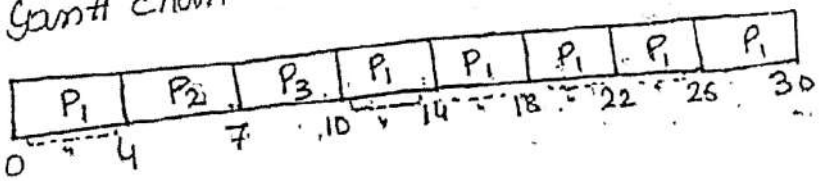
vi) We keep the ready queue as a FIFO queue. New process are added at the tail of the ready queue.

For eg:-

Process	Burst time
P_1	24 (20)(16)(12)(8)(4)
P_2	3
P_3	3

quantum time = 4ms

Gantt chart



Waiting time calculation.

Waiting time for P_1 = 26 − 20 = 6

P_2 = 4 − 0 = 4

P_3 = 7 − 0 = 7

17.

∴ Average waiting time = 17/3 = 5.6 secs.

Turn around time for P_1 = 30

P_2 = 7

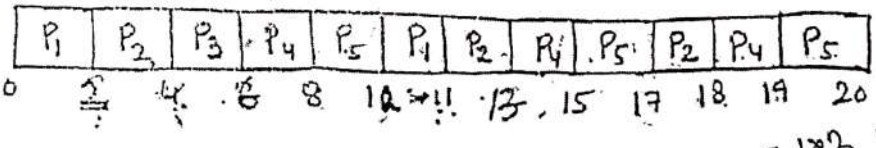
P_3 = 10

47/3 =

Process	Burst time
P_1	3
P_2	5
P_3	2
P_4	5
P_5	5

Quantum time = 2 ms

Gantt chant



$110 - 1 \times 2 = 8$

Waiting time for $P_1 = (10 - 2) = 8$

$P_2 = (17 - 4) = 13$

$P_3 = (4 - 0) = 4$

$P_4 = (18 - 4) = 14$

$P_5 = (19 - 4) = 15$

∴ A. w. t. = $54/5 = 10.8$ ms

Turn around time for $P_1 = 11$

$P_2 = 18$

$P_3 = 6$

$P_4 = 19$

$P_5 = 20$

$\therefore$ A. T. time $= 74/5 = 14.8$ ms

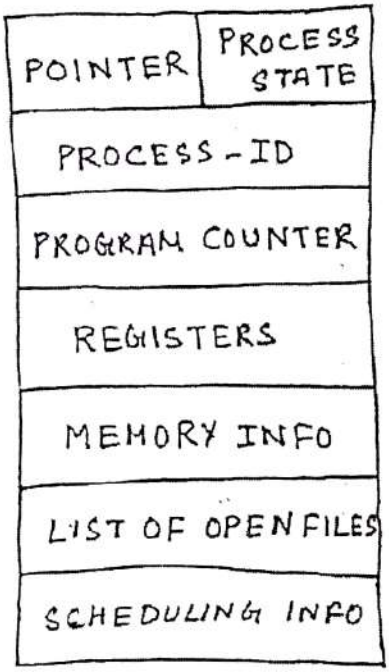
ADP

12.8.11

Short note on PCB or TCB (Task Control Block)

Full form of PCB (Process Control Block). Each process is represented in the operating system by a process control block. It is also called task control block.

The structure of PCB is given below



1) Process State – The state may be new, ready, running, waiting, terminated.

2) Pointer – It provides the link of next process It is useful during waiting for execution

or waiting for some event in a queue.

3) Program Counter – The counter indicate the address of the next instruction to be executed for this process

4) CPU Register – The register vary in number and type depending on computer architecture. The register may be accumulator, index register, general purpose register and so on.

5) CPU scheduling info – This information include a process priority and any other scheduling parameter.

6) Memory management info – This information include the content of base and limit register, the page table and so on.

7) Accounting information – This information include the amount of CPU and real time used, time limit, account no. and so on.

8) I/O information – The information include list of I/O devices, list of open files and so on.

Describe different criteria of scheduling

The criteria of scheduling includes the following -

1) CPU utilization - We want to keep the CPU as busy as possible. CPU utilization may range from 0 - 100% but in real system it ranges from 0 - 40% and for heavily used system it is upto 90%.

2) Throughput - Number of process executing per unit time is called throughput.

3) Waiting time - It is the sum of the time spend waiting in the ready queue.

4) Turn around time - The interval from the time of submission of the process to the time of completion is the turn around time Turn around time is the sum of the time spend in waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.

5) Response time - It is the time from submission of the request until the fast response is produced. The response time is the amount of time a process takes to start responding.

To make CPU scheduling a better and efficient technique we need to maximise the following and minimise the following.

Maximise - CPU utilisation, throughput
Minimise - Response time, waiting time, turn around time.

Disk Scheduling

What is disk scheduling? Describe different algorithm of disk scheduling.

ans:- Disk provide the bulk of secondary storage for computer system. Most disk drive are large one dimensional array of logical block where logical block is the smallest unit of transfer.

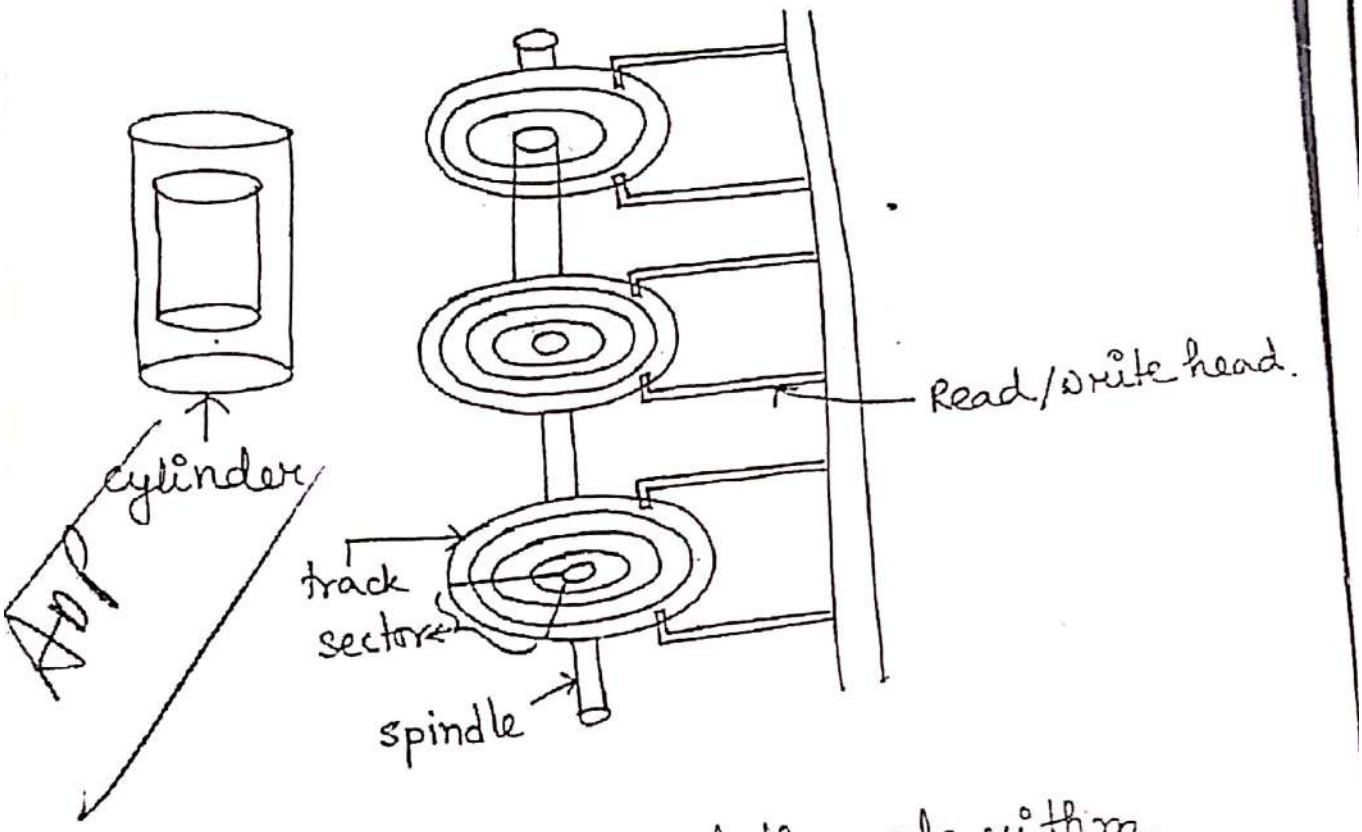
One of the responsibility of operating system is to use the computer hardware efficiently. For performing the disk operation there are certain disk scheduling algorithm.

Seek time - It is a time for the disk arm to move the head to the cylinder containing the desired sector.

Rotational latency - It is the additional time waiting for the disk to rotate the desired sector to the disk head.

Bandwidth – The disk's bandwidth is the total number of byte transferred.

The organization of disk is given below –



Different disk scheduling algorithm

There are many disk scheduling algorithm. Some of them are given below:-

1) FCFS – i) Full form of FCFS is First Come First Serve.

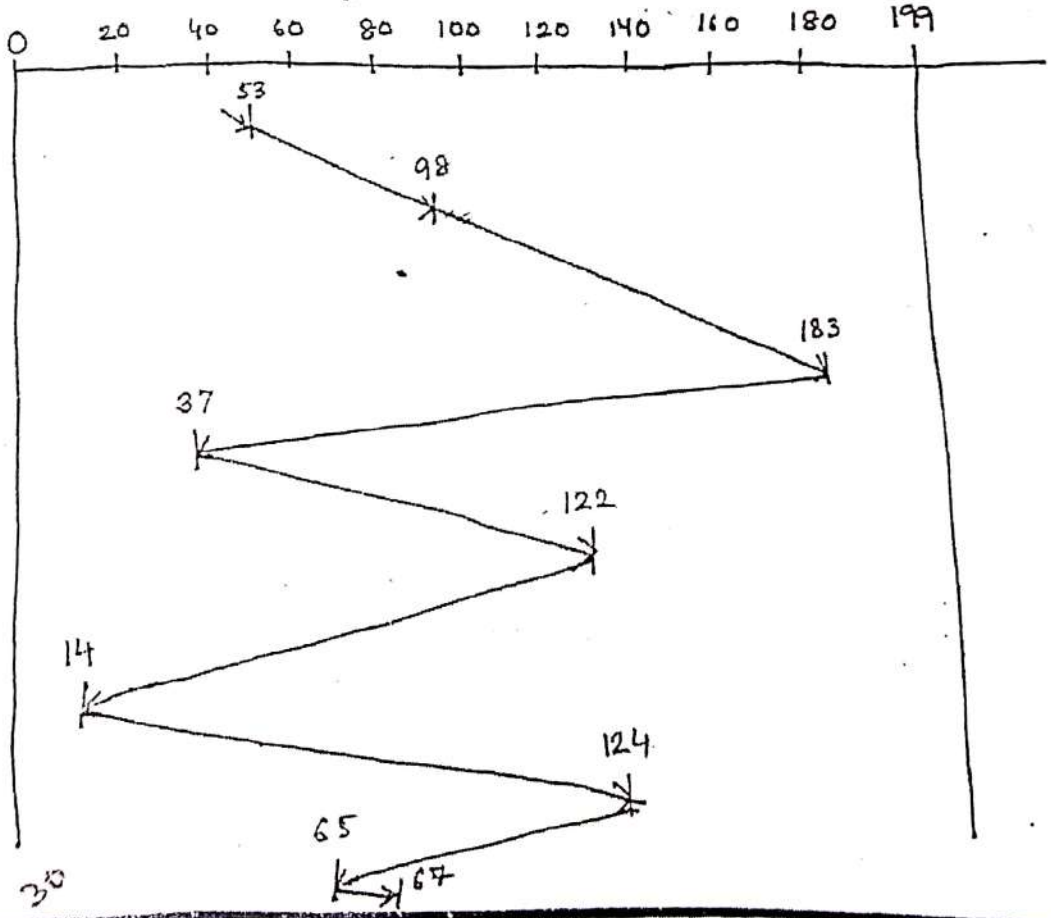
ii) It is one of the simplest disk scheduling algorithm.

iii) The criteria of this algorithm is that those cylinder arrive first will be served first.

iv) This is mainly First Come First Serve basis scheduling algorithm but it is not one of the best disk scheduling algorithm.

For eg:- Consider the following request for cylinder. 98, 183, 37, 122, 14, 124, 65, 67

Initially disks are at cylinder 53. Total number of disk is 200.



Total head movement = (98 − 53) + (183 − 98)
+ (183 − 37) + (122 − 37) + (122 − 14) +
(124 − 14) + (124 − 65) + (67 − 65)

= 45 + 85 + 146 + 85 + 108 + 110
+ 59 + 2

= 640

Average head movement = 640/8
= 80

2) SSTF - i) Full form of SSTF is Shortest Seek Time First.

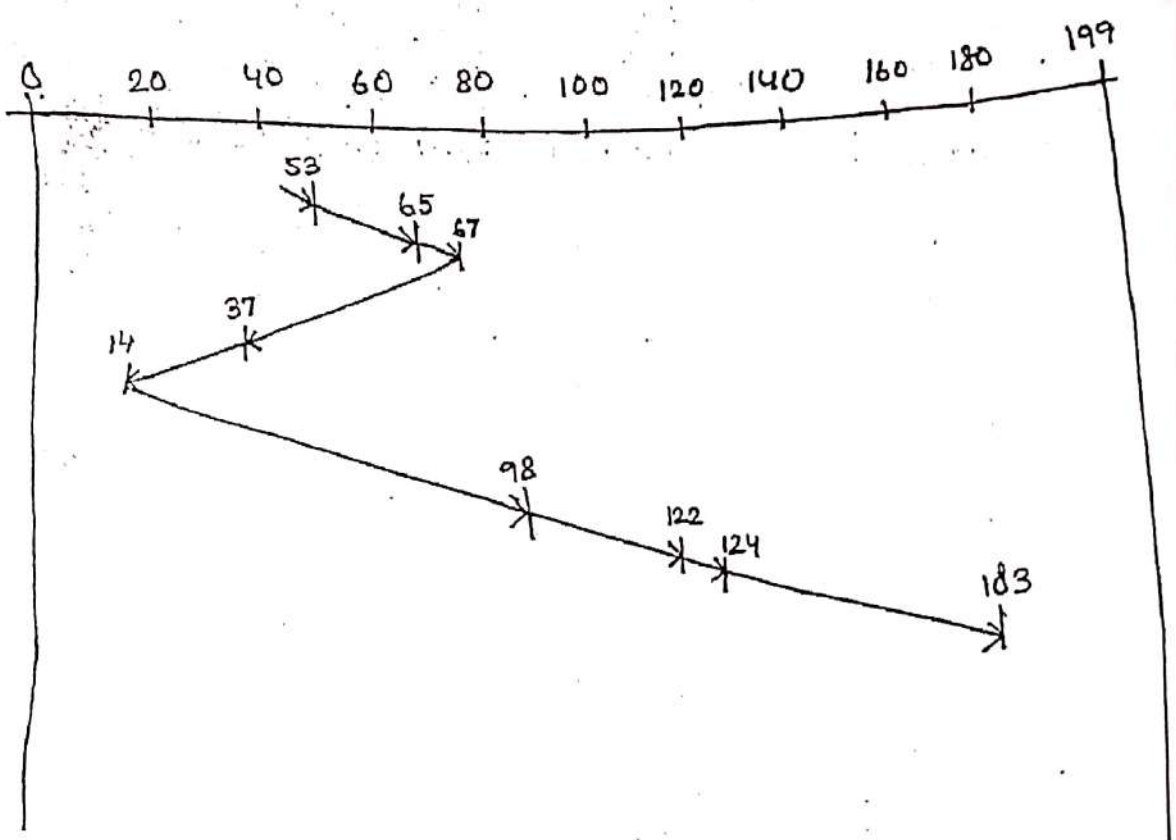
ii) SSTF algorithm select the request with the minimum seek time from the current head position.

iii) Since seek time increases the number of cylinder traversed by the head, SSTF select the pending request closest to the current head position.

Consider an example:

Suppose disk queue is 98, 183, 37, 122, 14, 124, 65, 67.

Initially disk are at 53.



Total head movement = (65-53) + (67-65)
+ (67-37) + (37-14) + (98-14) +
(122-98) + (124-122) + (183-124)

= 236

∴ Average head movement = 236/8

= 29.5

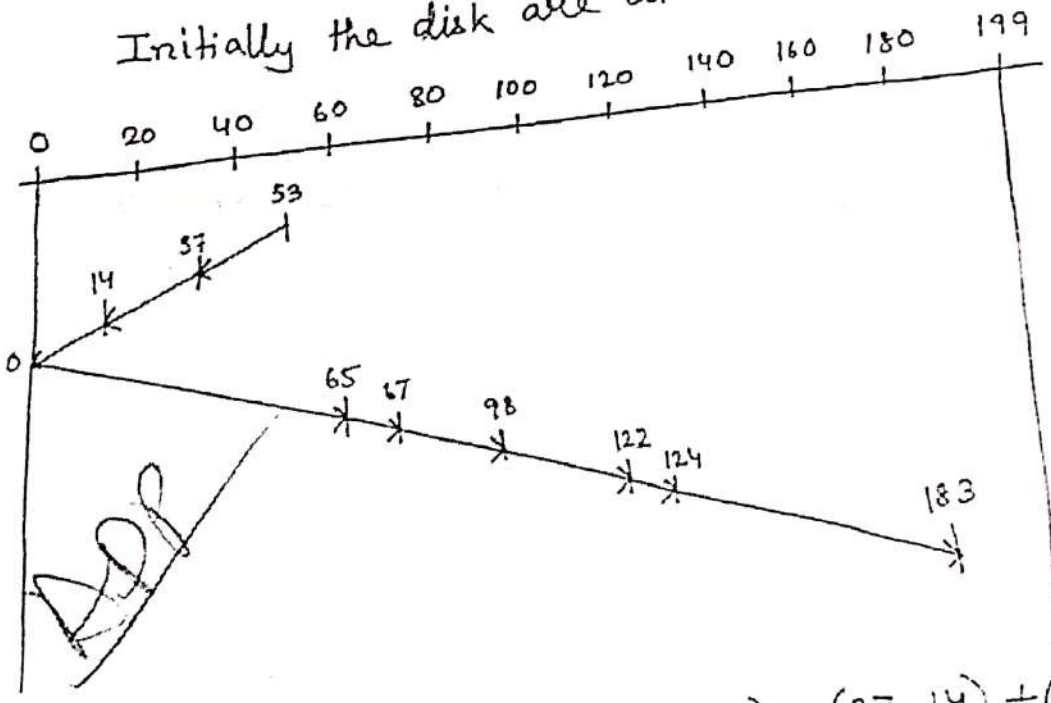
3) Scan scheduling algorithm – In scan scheduling algorithm the disk arm start at one end and moves towards the other

end, servicing request as it reaches each of its cylinder until it gets to the other end of the disk.

At the other end the direction of head movement is reverse and servicing continue. The head continuously back and forth across the disk.

For eg:- Suppose disk queue is 98, 183, 37, 122, 14, 124, 65, 67.

Initially the disk are at 53



Total head movement = (53-37) + (37-14) + (14-0)
+ (65-0) + (67-65) + (98-67) + (122-98)
+ (124 + -122) + (183 - 124)
= 16 + 23 + 14 + 65 + 2 + 31 +

Average head movement = 236/9
= 26.22.

Suppose that a disk drive has 5000 cylinder number from 0 to 4999. ~~Initial~~ The drive is currently servicing a request at cylinder 143 and previous request was 125. The disk queue is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. Apply FCFS, SSTF and Scan.

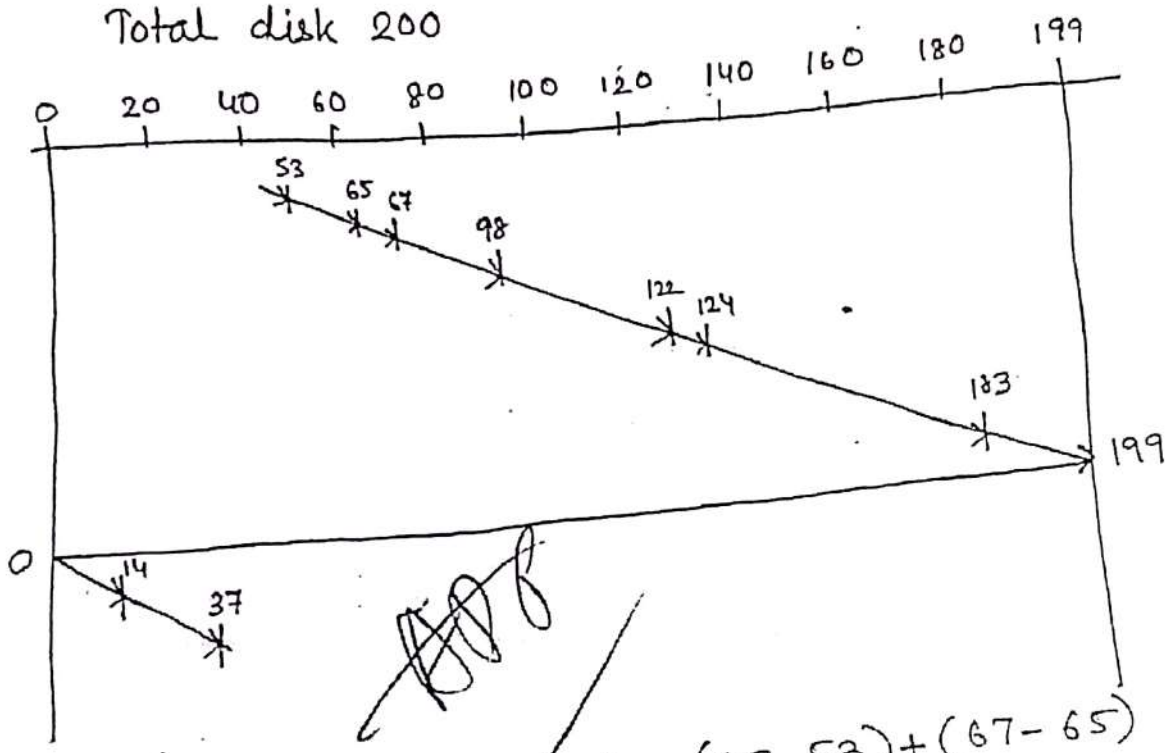
4) C_SCAN scheduling algorithm – Full form of C_Scan is circular scan scheduling algorithm. It is a variant of scan scheduling algorithm. It is designed to provide more uniform wait time. C_Scan moves the head from one end of the disk to the other end, servicing request along the way. When the head reaches to the other end it immediately return to the beginning of the disc without servicing any request on the return trip.

For eg:-

Disk queue: 98, 183, 37, 122, 14, 124, 65, 67

Initially at 53.

Total disk 200



Total head movement = (65-53) + (67-65)
+ (98-67) + (122-98) + (124-122)
+ (183-124) + (199-183) + (199-0)
+ (37-14) + (14-0)

= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 14 + 23 + 199 *

= 382

AHM = 382/10 = 38.2.

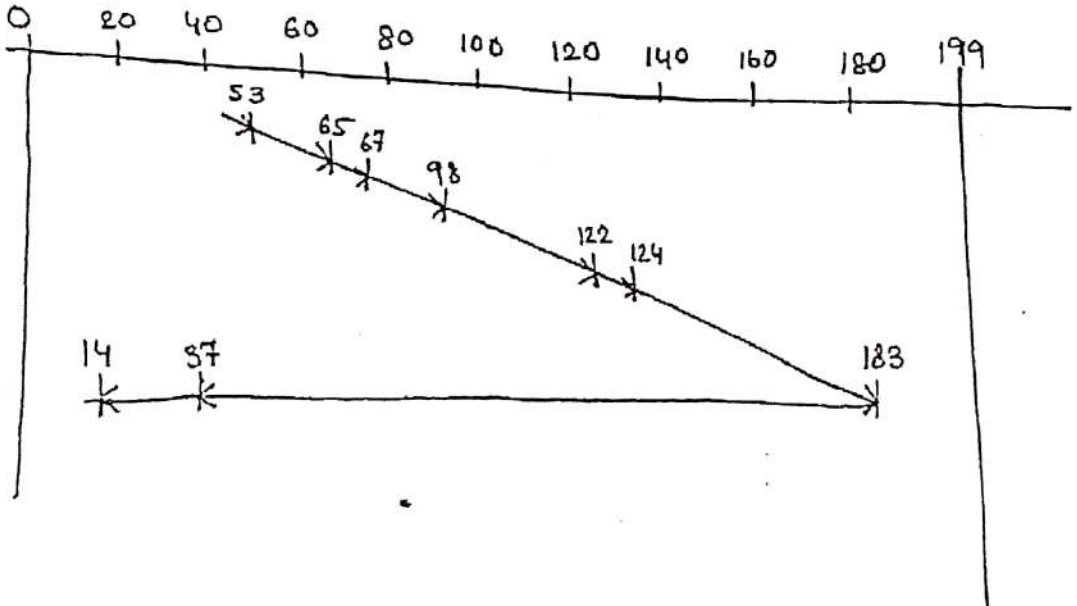
35

5) Look Scheduling algorithm - Head movement start from one direction to another direction servicing the request one by one. After servicing the last request at the one end it moves to next request directly.

For eg:- Disk queue : 98, 183, 37, 122, 14, 124, 65, 67.

Initially at 53

Total disk = 200



Total head movement = (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (183-37) + (37-14)

= 12 + 2 + 31 + 24 + 2 + 59 + 146 + 23

= 299

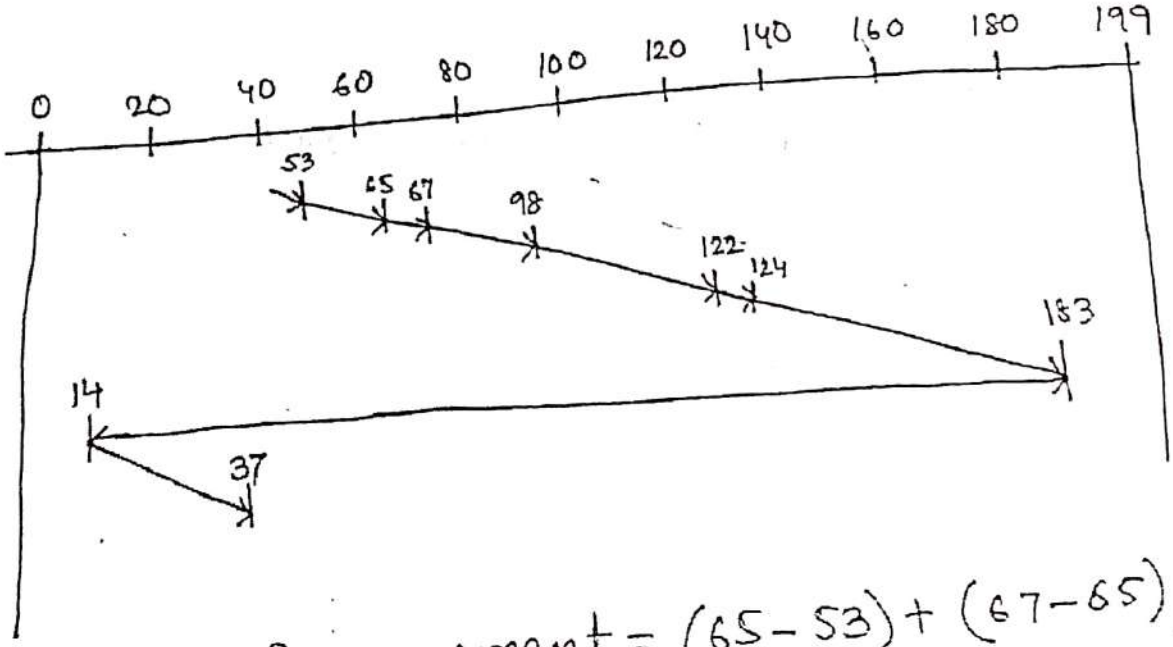
Average

Scanned by CamScanner

nt 6) C-Look scheduling algorithm - Full form of C-Look is circular Look. In circular look the direction of head movement is always same. Initially the head moves from one end to another end. After servicing last request it moves to the other end and then servicing continue.

For eg:- Disk queue : 98, 183, 37, 122, 14, 124, 65, 67.

initially at 53
Total disk 200



Total head movement = (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (183-14) + (37-14)

= 12 + 2 + 31 + 24 + 2 + 59 + 169 + 23

= 322

Average head movement = ~~40.25~~ 322/8

= 40.25

Scanned by CamScanner

Process Synchronization

1) What is cooperating process?

2) What is race condition?

*3) What is critical section? What are the criteria of critical section?

*4) What is semaphore? How semaphore can be implemented?

5) Describe the followings:

a) Reader-~~xi~~ writer problem

b) Dining philosopher problem

c) Producer consumer problem

d) Monitor

6) What is thread? What are the different types of thread? Difference between process and thread.

Process Synchronization

6.8.11

1) What is cooperating processes?

A cooperating process is one that can affect or be affected by other processes executing in the system. Cooperating processes may either directly share a logical address space or be allowed to share data only through files.

2) What is race condition?

A situation where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which access take place is called race condition.

To guard against race condition we need to ensure that only one process at a time can be manipulating the common variable. To make such a guarantee we require some form of synchronization of the processes.

3)/ What is critical section? What are the criteria of critical section?

Consider a system consisting of n processes $P_1, P_2, \ldots P_n$. Each processes have a common segment called critical section in which the process may be changing common variables, upgrading a table, writing a file and so on. The execution of critical section by the processes is mutually exclusive. The critical section problem is to design a protocol that the processes can used to cooperate.

Each ~~critical section~~ process consist of -

i) Entry section
ii) Critical section
iii) Exit section
iv) Remainder section

The structure of each process is given below

~~do~~

```
do
{
  entry section
  Critical section
  Exit section
  Remainder section
}
while(1);
```

APP

Requirement for critical section :-

i) Mutual Exclusion - If process P_i is executing in its critical section no other processes can be executing in their critical section.

ii) Progress - If no processes is executing in its critical section and some processes wish to enter their critical section then only those processes that are not executing in their remainder section can participate in the decision on which will enter its critical section next.

iii) Bounded waiting - ~~Here~~ There exists a bound on the number of times the process

is allowed to enter their critical section after a process has made a request to enter its critical section and before that request is granted.

Example of critical Section.

Consider a railway reservation system which is centralized. Any person from anywhere from any place to any place can get the ticket.

Suppose two persons from different station want to reserve their ticket. The train number and destination of the train is common. The two persons try to get the reservation at the same time. Unfortunately there is only one seat is available. Therefore both are trying for the seat. This is also called critical section problem.

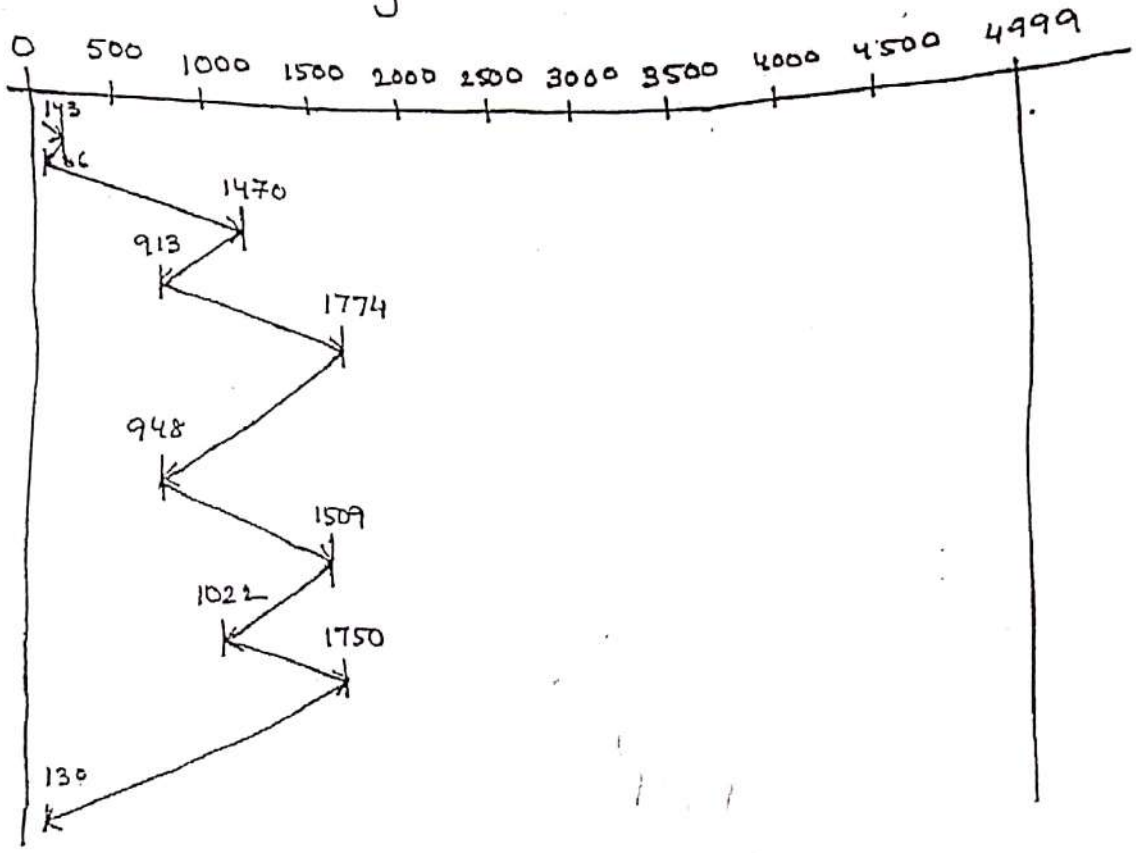
The solution for the problem is when one process is executing in its critical section, no other process is to be allowed to execute in its critical section.

APP

H/W

Total disk = 5000 FCFS

Initially at 143



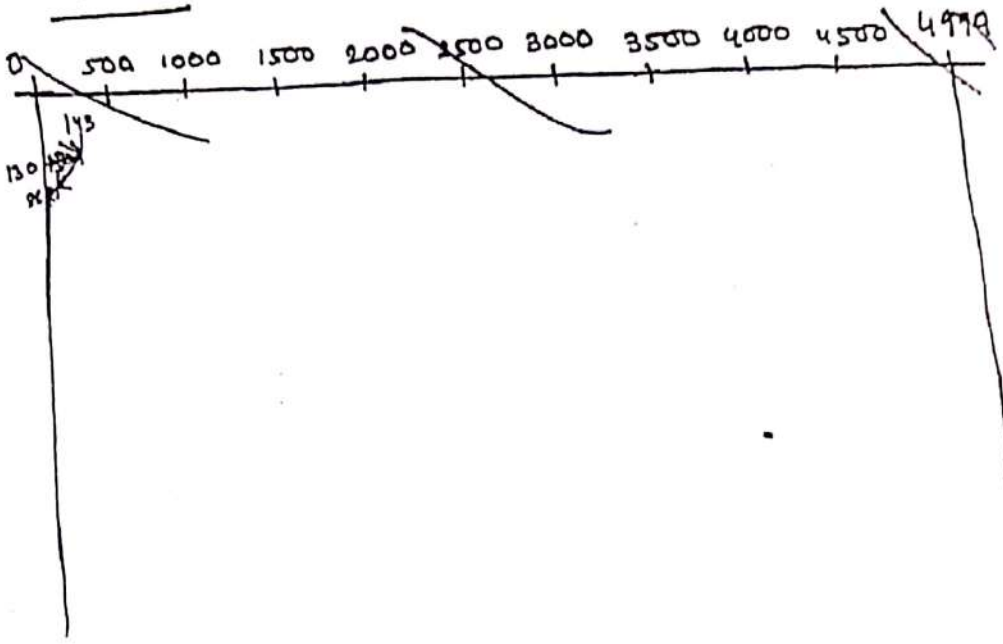
Total head movement = (143-86) + (1470-86)
+ (1470-913) + (1774-913) + (1774-948)
+ (1509-948) + (1509-1022)
+ (1750-1022) + (1750-130)

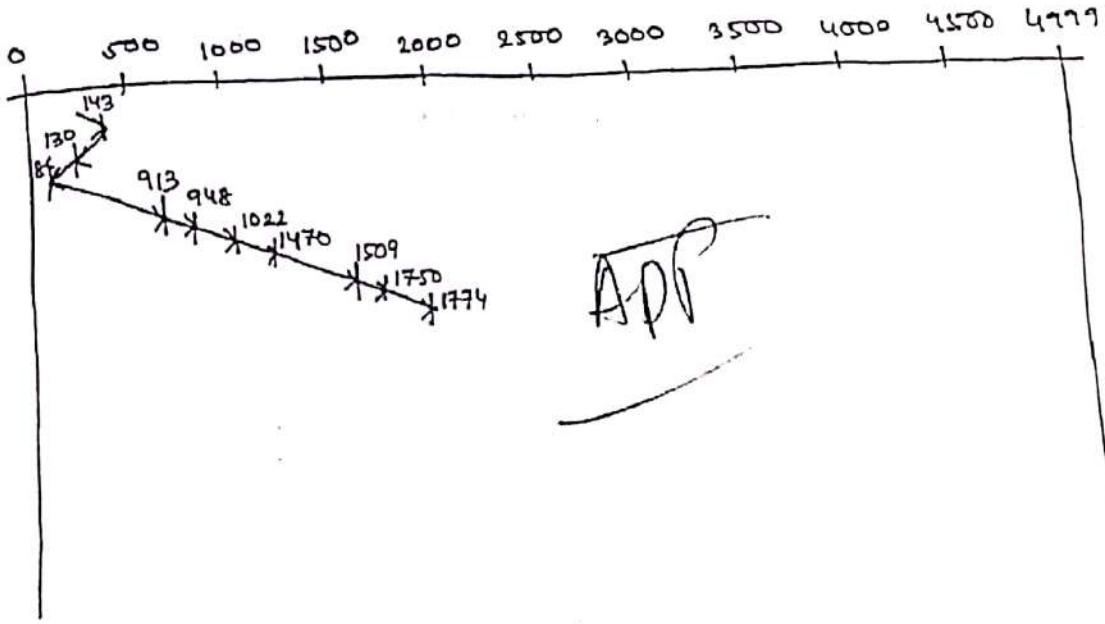
= 57 + 1384 + 557 + 861 + 826 + 561
+ 487 + 728 + 1620

= 7081

∴ Average head movement = 7081/9
= 786.7

SSTF

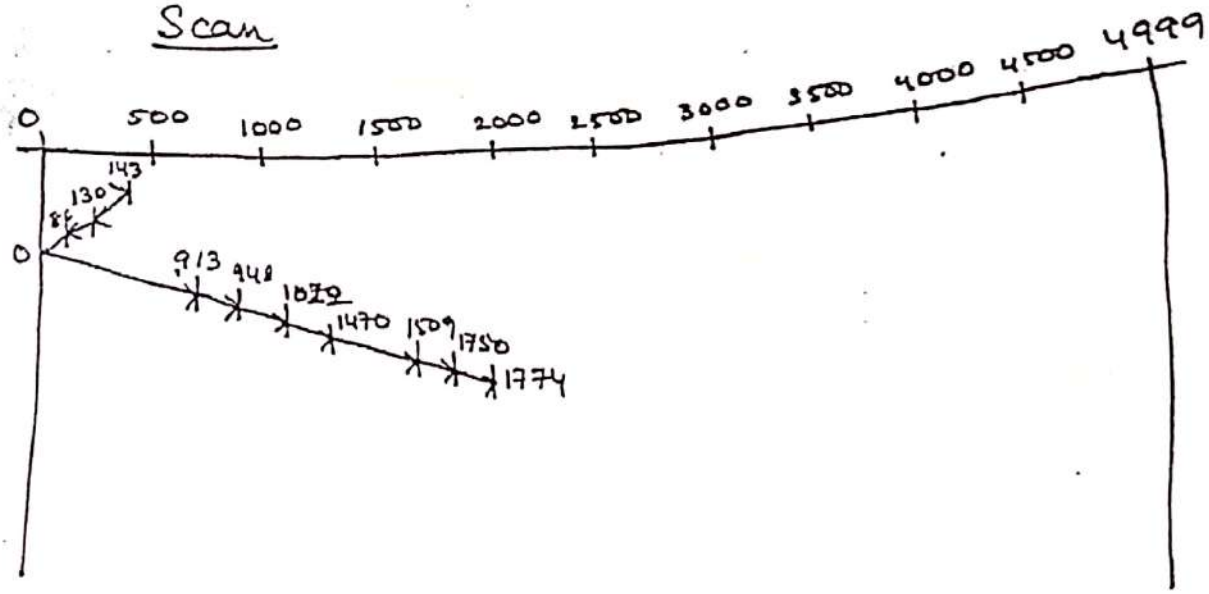




948)

∴ Total head movement =

Scan



Total head movement =

9.8.11

What is semaphore? How semaphore can be implemented?

ans:- The selection to the critical section problem can be implemented to the help of semaphore. Semaphore is a process synchronization tool. A semaphore S is an integer variable which contains (+)ve value. Semaphore is implemented with the help of two atomic operations.

1. wait()
2. signal()

The structure of wait() is given below:-

```
wait(s)
{
while(s<=0)
; // do no operation
s=s-1;
}
```

The function of signal() is given below:

```
signal(s)
{
  s=s+1;
}
```

~~Modify~~

Modification to the integer value of semaphore in the wait() & signal() operation must be

Modification to the integer value of semaphore in the wait() & signal() operation must be executed indivisibly ie, when one process modifies semaphore no other processes can simultaneously modify that semaphore.

The implementation of semaphore is given below by the following programming segment.

```
typedef struct
{
  int value;
  struct process *L;
} semaphore;
semaphore s;
void wait (semaphore s)
{
 s.value--;
 if (s.value <0)
  {
    add this process to s.L;
    block();
  }
}
void signal (semaphore s)
{
  s.value++;
```

```
if (s.value ≤ 0)
 {
   Remove a process from S.L ;
   wakeup(P);
 }
}
```

Producer consumer problem

There are set of processes. some of them produces some data item. There are another set of processes who consumes the data item. There is a fixed size buffer where producer produces and places the data item. Consumer consume the data from the fixed size buffer. Producer and consumer operate concurrently. Therefore there is a possibility of starvation for both the producer and consumer. When producer produces some data item but consumer never consume it therefore producer can starve. Similarly consumer can ~~start~~ when there are no data item available for

consuming.

The implementation of producer and consumer problem with the help of semaphore:

structure of producer

```
do
{
  produce an item
  wait (empty);
  wait (mutex);
    use the data item;
  signal (mutex);
  signal (full);
} while (1);
```

structure of consumer

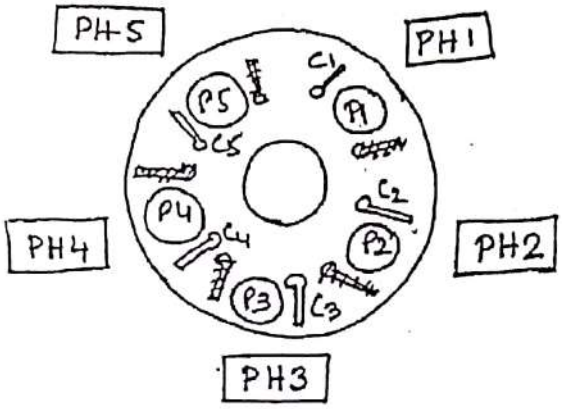
```
do
{
  wait (full);
  wait (mutex);
   Remove the data item;
  signal (mutex)
  signal (full);
} while (1);
```

Dining philosopher problem

Consider five philosopher who spend their lives thinking and eating. The philosopher share a common circular table surrounded by five chairs each belonging to one philosopher.

In the centre of the table there is a bowl of rice and the table is laid with 5 single chopsticks. When a philosopher thinks he does not interact with his colleagues. From time to time a philosopher gets hungry and tries to pick up the two chopsticks that are closest to him. A philosopher may pick only one chopstick at a time but he cannot pick up a chopstick that is already in the hand of neighbour. When a hungry philosopher has both his chopsticks at the same time he eats without releasing the chopsticks. When he finished eating he puts down both the chopsticks and start

thinking again.



Here PH1, PH2, PH3, PH4, PH5 are philosopher. P1, P2, P3, P4, P5 are plate. C1, C2, C3, C4, C5 are chopsticks.

The implementation of dining philosopher using semaphore is given below:-

```
do
{
    wait (chopstick [i]);
    wait (chopstick [(i+1)% 5]);
    // eating
    signal (chopstick [i]);
    signal (chopstick [(i+1)% 5]);
} // thinking
while (true);
```

* Reader Writer Problem

A data object such as file or record is to be shared among several concurrent processes. Some of these processes may want only to read the content of the shared object, whereas others may want to update the shared object. The first type of processes that are interested only in reading called reader and the remaining processes are called writer.

If two readers access the shared data object simultaneously no adverse effect will result. However if a writer and some other process access the shared object simultaneously chaos may ensured.

There are two types of reader writer problem. In the first reader writer problem no reader will be kept waiting unless a writer is waiting. In this case writer

may starve. In second reader writer problem once a writer is ready, that writer performs as soon as possible.

The implementation of reader writer problem using semaphore is given below -

```
Semaphore wrt, mutex ;
int readcount ;
```

Structure of writer process

```
wait (wrt)
// writing is performed
signal (wrt);
```

Structure of reader process

```
wait (mutex);
readcount ++ ;
if (readcount == 1)
    wait (wrt);
signal (mutex);
// Reading is performed
wait (mutex);
```

```
readcount--;
If (readcount == 0)
    signal (wrt);
signal (mutex);
```

DeadLock

1) What is deadlock ? What are the necessary conditions of deadlock ? ~~Short note~~

2) Short note on resource allocation graph.

3) Describe different methods of controlling deadlock.

4) What is safe and unsafe state ?

5) Describe banker algorithm. One mathematical problem ~~from~~ on banker algorithm.

6) What do you mean by recovery from deadlock ?

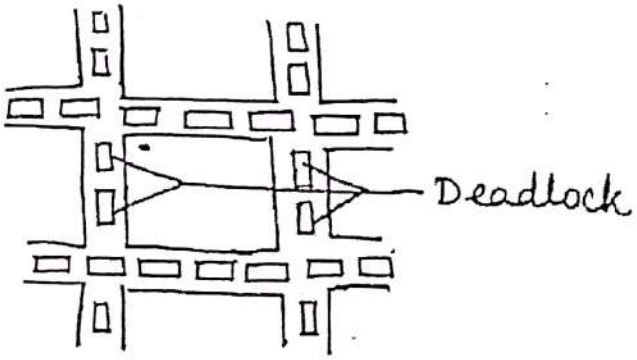
In a multiprogramming environment several processes may compete for a finite

number of resources. A process request resources if the resources are not available at that time the process enter a wait state. Waiting processes may never again change ~~it~~ state because the resources they have requested ~~or~~ are held by other waiting processes. This situation is called deadlock.

Example of deadlock:-

i) Two trains moving in the same track from the opposite direction.

ii) Traffic jam in a four level crossing.



Necessary conditions of deadlock

A deadlock situation may arise if the following four conditions hold simultaneously in a system:-

i) Mutual exclusion
ii) Hold and wait
iii) No preemption
iv) Circular wait

i) Mutual exclusion - At least one resource must be held in a non sharable mode. ie, only one process at a time can use the resource. If another process request that resource the requesting process must be delay until the resource has been released.

ii) Hold and wait - A process must be holding atleast one resource and waiting to acquire additional resources that are currently being held by other processes.

iii) No preemption - Resources cannot be preempted ie, resource can be released only when that process has completed its task.

iv) Circular wait - A set $\{P_0, P_1, P_2 \ldots P_{n-1}, P_n\}$ of waiting processes must exist such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, P_2 is waiting for a resource that is held by P_3, ... P_{n-1} is waiting for a resource that is held by P_n, P_n is waiting for a resource that is held by P_0.

Short note on resource allocation graph

Deadlock can be described more precisely in term of directed graph called resource allocation graph.

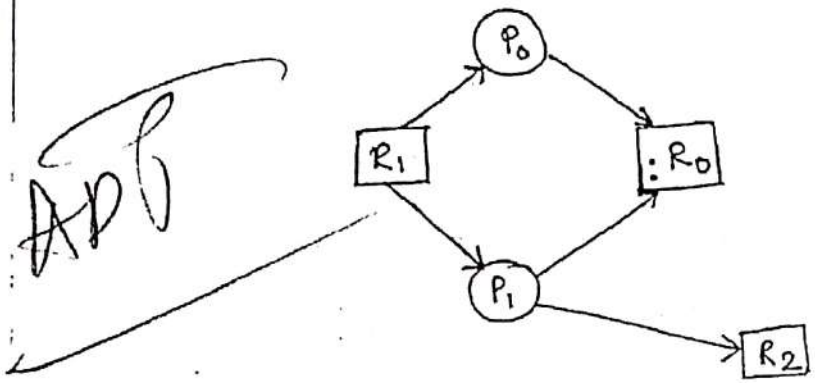
A resource allocation graph consist of set of vertices and set of edges. Vertices are represented by either circle or rectangle. Processes are represented as circle, resources are represented as rectangle. If a resource has more than one type then it is called

instance. For eg:- ~~A fo~~ Suppose a system consist of 5 printers. Then the instance of printer is 5. There are 2 types of edges-

i) Request edge

ii) Assigned edge

For eg:-



RAG

$G = (V, E)$

$V = \{P_0, R_1, R_0, P_1, R_2\}$

E consist of

i) Request edge

$\{P_0, R_0\}$

$\{P_1, R_2\}$

$\{P_1, R_0\}$

16.8.11

Methods for controlling deadlock

We can deal the deadlock problem in one of the three way

i) We can use a protocol to prevent or avoid deadlocks ensuring that the system will never enter a deadlock state.

ii) We can allow the system to enter a deadlock state, detect it and recover

iii) We can ignore the problem and consider that the deadlock never occur in the system. This solution is used by most operating system including unix.

To ensure deadlock never occur the system can use either a deadlock prevention or deadlock avoidance.

Deadlock prevention is a set of methods for ensuring that atleast one of the necessary condition cannot hold. Deadlock avoidance requires that the operating system be given in advance additional information

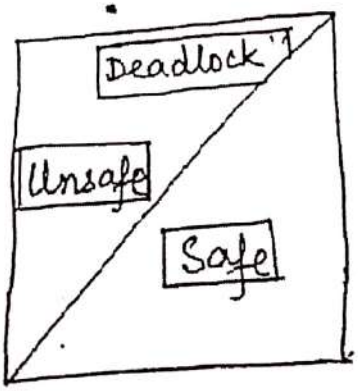
concerning which resources a process will request and use during its lifetime.

What is safe and unsafe state. Write the Banker algorithm with example.

Safe state - A state is safe if the system can allocate resources to each process in some order and still avoid a deadlock

~~The~~ A system is in safe state if there exist a safe sequence.

Unsafe state - A set of processes is called unsafe if there exist a deadlock.



Banker algorithm

Banker algorithm is one of the deadlock avoidance algorithm. It is mainly used in banking system and is implemented in a deadlock of processes. Banker algorithm consist of the following data structure:-

i) ~~Avail~~ Available – A vector of length m indicate the number of available resources of each type. If Available[j] = k then there are k instances of resource type R_j. For example Available[3] = 5 means resource R_3 has 5 instances.

ii) Max – It is an n×m matrix that defines the maximum demand of each process. If max[i,j] = k means process P_i may request atmost k instances of resource type R_j. For example

MAX[2,5] = 9 means process P_2 can demand 9 instance of resource type R_5.

iii) Allocation - A n x m matrix defines the number of resources of each type currently allocated to each process. If Allocation [i,j] = k means process P_i is allocated k number of instances of resource type R_j. For example Allocation [2,5] = 6 means P_2 process allocated 6 instances of resource type R_5.

iv) Need - A n x m matrix that indicate the remaining resources of each process need [i,j] = Max [i,j] - Allocated [i,j].

Banker algorithm consist of two parts

i) Safety algorithm

ii) Resource request algorithm

Safety algorithm

This algorithm find out whether or not a system is in safe state or not.

step1 :- Let Work and Finish be two vectors. of length m and n respectively.
Initially Work = Available
Finish[i] = False $i = 1$ to n

step2 :- Find an i such that Finish[i]=False and Needi $\leq$ Work.
If no such i exist go to step4.

step3 :- Work = Work + Allocationi
Finish [i] = True
Go to step 2.

step4 :- If Finish [i] = True for all i then the system is in safe.

Resource Request Algorithm

Let Requesti be the request vector for process P_i
Request [i, j] = k means process P_i can request k number of instances of resource type R_j

Solu

step 1:- If $Request_i \leq Need_i$, go to step 2.

step 2:- If $Request_i \leq Available$ go to step 3

step 3:- $Available = Available - Request_i$

$Allocation_i = Allocation_i + Request_i$

$Need_i = Need_i - Request_i$

Consider a system with 5 processes P_0, P_1, P_2, P_3, P_4 with resource type A, B, C. A consist of 10 instances, B consist of 5 instances, C consist of 7 instances. Consider the following snapshots.

Process	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			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Solⁿ.

Work = Available = 3 3 2

Finish[i] = False i = 1 to 5

Finish[1] = False

Finish [2] = False

Finish[3] = False

Finish [4] = False

Finish [5] = False

Need

	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

When i = 1

Finish[1] = false yes

743 ≤ 332 No

When i = 2

Finish[2] = False yes

122 ≤ 332 yes

Work = 332

~~Finish [2]~~ = 332 + 200

= 532 Finish[2] = true

when $i = 3$

Finish[3] = ~~True~~ False Yes

600 ~~743~~ $\leq$ 532 No

when $i = 4$

Finish[4] = False Yes

011 $\leq$ 532 Yes

Work = 532 + 211

= 743

Finish[4] = True

when $i = 5$

Finish[5] = False Yes

431 $\leq$ 743 Yes

Work = 743 + 002

= 745

Finish[5] = True

when $i = 1$

Finish[1] = False Yes

743 $\leq$ 745 Yes

Work = 745 + 010

= 755 Finish[1] = True

68

When i = 3

Finish [3] = False Yes

600 ⩽ 755 Yes

Work = 755 + ~~902~~ 302

= 1057

Finish [3] = True.

The above processes are in safe because there exist a safe sequence. The safe sequence is given below :-

P_1, P_3, P_4, P_0, P_2

23.8.11

✓ Deadlock Prevention

For a deadlock to occur there are 4 necessary condition must hold. While ensuring that atleast one of this condition cannot hold we can prevent the occurence of deadlock.

1) Mutual exclusion – The mutual exclusion must hold for non sharable devices.

Example – Printer can't be simultaneously share by several processes, sharable resources on the other hand do not require mutually exclusive access and thus cannot involve in deadlock.

Therefore we cannot prevent deadlock by avoiding mutual exclusive condition.

2) Hold and wait – To ensure that hold and wait condition never occur in the system. We must guaranteed that whenever a process request resource it does not

There are many protocols:

1) One protocol ∈ that can be used requires each process to request and be allocated all its resources before it begins execution.

2) An alternative protocol allows in process to request resources only when the process has none. A process may request some resources and use them. Before it can request additional resources it must release all the resources ie, currently allocated.

3) No preemption - To ensure that this condition does not hold we can use the following protocol. If a process is holding some resources and request another resource that cannot be immediately allocated, then all resources are preempted.

4) Circular wait - One way to ensure that this condition never hold is to impose a total ordering of all resource type and to require that request resources in an increasing order.

Recovery from deadlock

There are two ways to recover from deadlock.

i) Process termination - a) Abort all deadlock processes - This method clearly will break the deadlock cycle.

b) Abort one process at a time * until the deadlock cycle is eliminated.

ii) Resource preemption -

a) Selecting a victim - which resource and which process are to be preempted depends on some parameter.

b) Roll back - This method requires the system to keep more information about the state of all running processes.

Consider the following snapshots of the system

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
									1	5	2	0
P_0	0	0	1	2	0	0	1	2				
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Answer the following question:

a) What is the content of ~~matrix~~ max need?

b) Is the system is in safe?

c) If the a request from process P_1 arrives for (0 4 2 0). can the request be granted ? Immediately ?

Work = Available = 1 5 2 0

Finish [i] = False i = 1 to 5

Finish [1] = False

Finish [2] = False

Finish [3] = False

Finish [4] = False

Finish [5] = False

Need

	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

When i=1

Finish[1] = False Yes

0000 ≤ 1520 Yes

Work = 1520 + 0012

= 1532 & Finish[1] = True.

when i=2

Finish[2] = False Yes

0750 ≤ 1532 No

when i=3

Finish[3] = False Yes

1002 ≤ 1532 Yes

Work = 1532 + 1354

= 2886 Finish[3] = True

when i=4

Finish[4] = False Yes

0020 ≤ 2886 Yes

Work = 2886 + 0632

= ~~3518~~ Finish[4] = True

= 214118

when i=5

Finish[5] = False Yes

0642 ≤ ~~3518~~ Yes

214118

Work = 2 14 11 8 + 8 0 1 4

= 2 14 12 12 Finish [5] = True

when i = 2

Finish[2] = False yes

7 5 0 ≤ 2 14 12 12 yes

Work = 2 14 12 12 + 1 0 0 0

= 3 14 12 12

The ~~abo~~ safe sequence

P_0, P_2, P_3, P_4, P_1

c) P_1 request

R_1 = 0 4 2 0

step 1 : $R_1 \leq$ Need

0 4 2 0 ≤ 0 7 5 0 Yes

step 2 : $R_1 \leq$ Available

0 4 2 0 ≤ 15 2 0 Yes

step 3 : Allocation$_1$ = Allocation$_1$ + R_1

= 1 0 0 0 + 0 4 2 0

= 1 4 2 0

$Need_1 = Need_1 - Request$

$= 0750 - 0420$

$= 0330$

$Available = Available - Request$

$= 1520 - 0420$

$= 1100$

Now, the following snapshots is

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	1	0	0
P_1	1	4	2	0								
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Need table is

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

75

When i = 1

Finish [1] = False Yes

0000 ≤ 1100 Yes

1112 Finish[1] = Tru

when i = 2

Finish [2] = false

0330 ≤ 1112 No

when i = 3

Finish[3] = false

1002 ≤ 1112 yes

Work = 1112 + 1354

= 2466

Finish[3] = True

when i = 4

Finish[4] = false

0020 ≤ 2466 yes

Work = 2466 + 0632

= 21098

Finish[4] = True

when i = 5

Finish[5] = false yes

0642 ≤ 21098 yes

Work = 21098 + 0014

= 2 10 10 12

Finish[5] = True

When i = 2
Finish[2] = false yes
0330 ≤ 2 10 10 12 yes

Work = 2 10 10 12 + 1 4 2 0

= 3 14 12 12

The safe sequence is P_0, P_2, P_3, P_4, P_1

Memory Management

26.8...

1) Difference between logical address space and physical address space.

2) How d logical address space is divided into Physical address space?

3) Describe different memory allocation scheme

4) What is fragmentation? Describe different types of fragmentation with example.

5) What is compaction?
What is first fit, best fit and worse fit?
Describe it with example.

6) What is paging? Describe paging hardware

7) What is segmentation? Difference between paging and segmentation.

8) Short note on the following:-

a) TLB

b) Inverted page table

c) Swapping

9) What is virtual memory? How virtual memory can be implemented with the help of demand paging?

10) What is page fault? Describe the technique of page fault.

11) What do you mean by page replacement? Describe FIFO, optimal, LRU page replacement algorithm.

12) Short note

a) Belady anamoly

b) Thrashing

Logical and physical address

Logical address space	Physical address space
i) Logical address space is generated by the CPU.	i) Physical address space is generated by the main memory.
ii) An address generated by the CPU referred to as logical address or virtual address. The set of all logical add-	ii) An address seen by memory unit ie., the one loaded into memory address register of the memory is known as

resses known as logical address space

physical address. The set of all ~~phy~~ physical addresses known as physical address space.

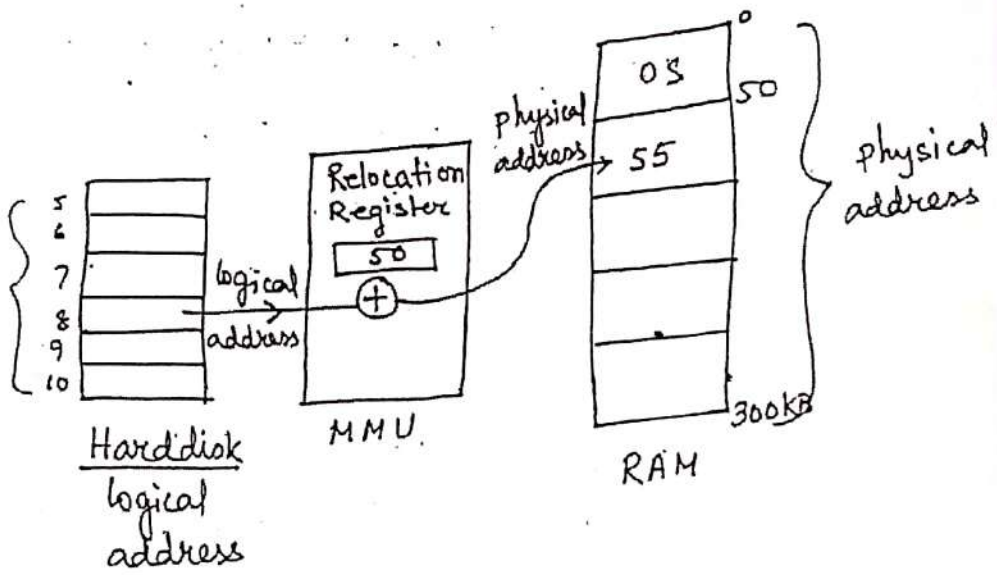
The compile time and ~~low~~ load time address binding generate identical, ~~and~~ logical and physical addresses but in execution time the logical and physical addresses are different.

There is a special hardware unit which is used to convert logical address into physical address. This unit is called MMU (Memory Management Unit)

Within memory management unit there is a special register called relocation register which store the base address of the memory.

Physical address = content of the relocation register + logical address

The following diagram shows how logical address is converted into physical address.



What is fragmentation? Different types of fragmentation.

Fragmentation is the wastage of memory during memory allocation of processes.

There are two types of fragmentation

i) Internal fragmentation

ii) External fragmentation

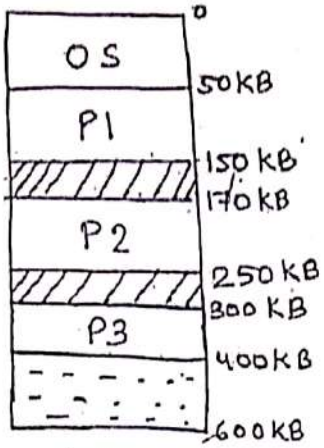
External fragmentation – processes are loaded and removed from memory, the free memory space is broken into little pieces.

External fragmentation exist when enough total memory space exist to satisfy a request but it is not contiguous. Therefore storage is fragmented into large number of small memory pieces.

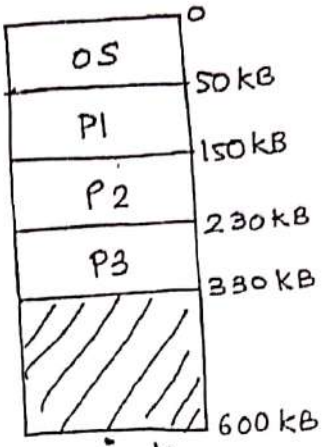
Internal fragmentation - In Internal fragmentation memory is internal to a partition but is not being used. Internal fragmentation occur due to the system.

Compaction is the technique of moving the all free space to one side of the memory and all processes to the another side of the memory. Compaction remove the problem of external fragmentation

Con Compaction recover wastage of memory spaces. Consider example.



Fragmentation



Compaction

What is first fit, best fit, worst fit?

First fit – Allocate the first memory space that is big enough. Searching can start from the beginning of the list.

Best fit – Allocate the smallest memory space that is big enough. Operating system search the entire list and allocate the most suitable memory space.

Worse fit – Allocate the largest memory space. Operating system search the entire list.

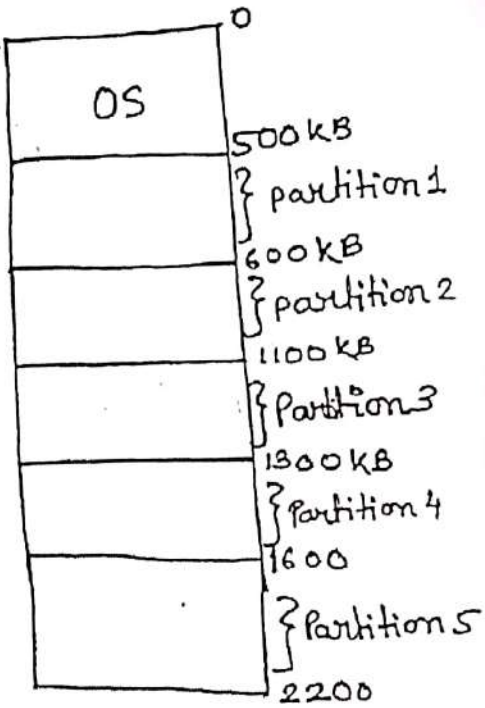
83

Example

Given memory partition of 100kB, 500 kB, 200 kB, 300 kB and 600 KB in order. Apply first fit, best fit & worst fit algorithm for the processes of ~~20~~: 212 KB, 417 KB, 112 KB, & 426 KB in order. Which algorithm makes most efficient use of memory.

First fit

Process	Allocation	External fragmentation
212 KB	500 KB	(500-212)= 288
417 KB	600 KB	(600-417)= 183
112 KB	200 KB	(200-112)= 88
426 KB (Not allocated)	×	×



Total External fragmentation - (288 + 183 + 88)
= 559

Internal fragmentation - (100 + 300) = 400 ~~kB~~

Best Fit

Process	Allocation	External fragmentation
212 KB	300KB	(300 − 212) = 88
417 KB	500KB	(500 − 417) = 83
112 KB	200 KB	(200 − 112) = 88
426 KB	600 KB	(600 − 426) = 174

External fragmentation = (88 + 83 + 88 + 174)
= 433

Internal fragmentation = 100

Worst fit

Process	Allocation	External fragmentation
212 kB	600 KB	(600 − 212) = 388
417 KB	500 KB	(500 − 417) = 83
112 KB	300 kB	(300 − 112) = 188
426 KB	X	X

External fragmentation = 388 + 83 + 188 = 659

Internal = 100 + 200 = 300

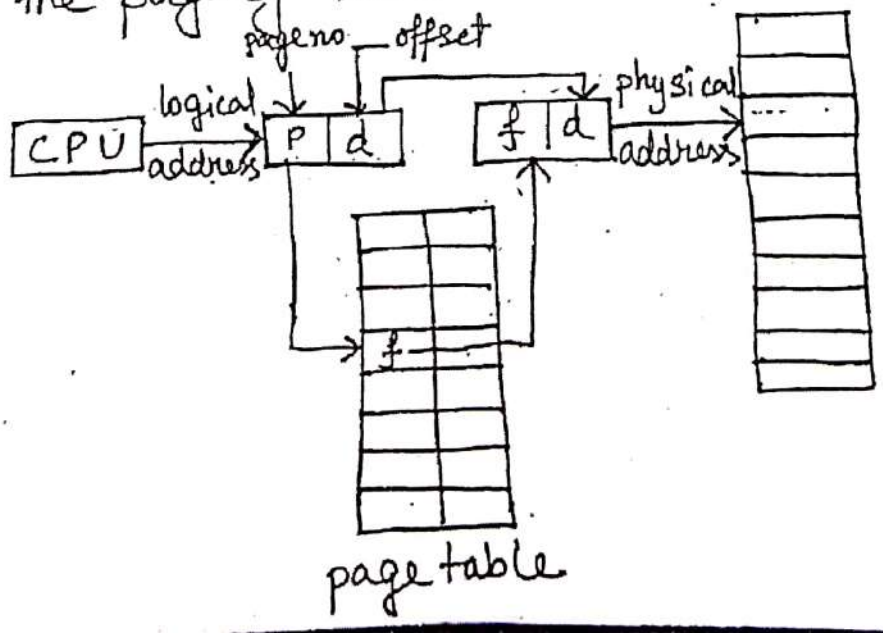
What is paging? Describe paging hardware scheme

Paging is a memory management ~~skill~~ that permits the physical address space of a process to be non contiguous. Paging avoids considerable problem of fitting the various sized memory chunks on to the hard disk.

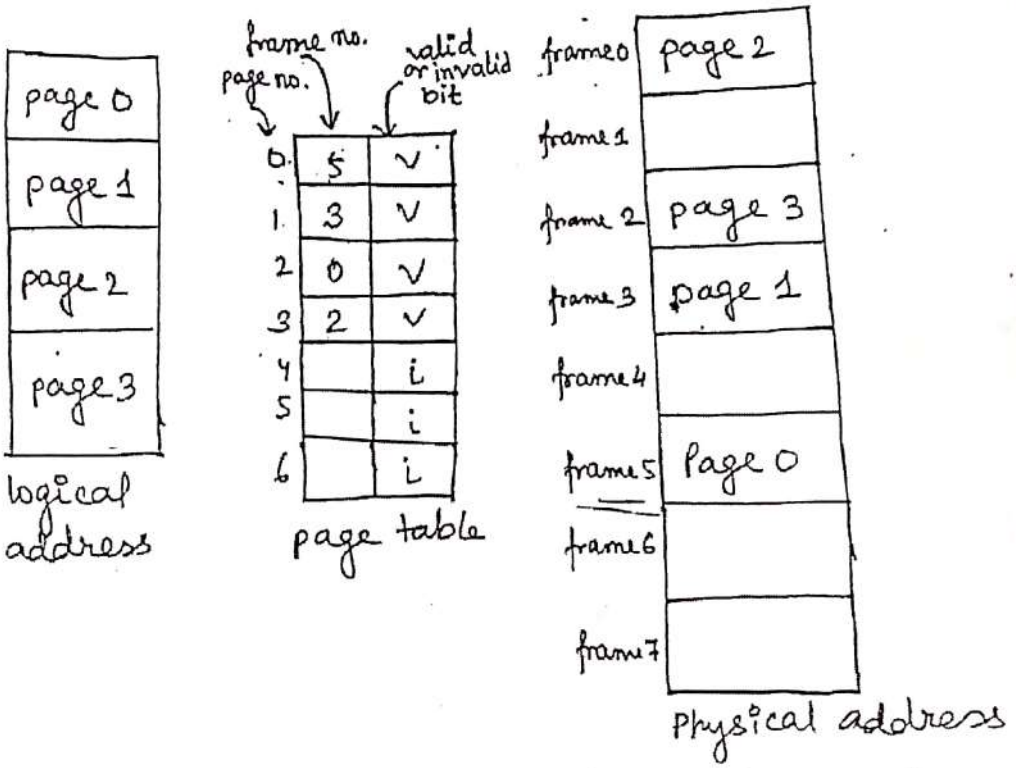
Physical memory is broken into fixed size partition called frame.

Logical address are divided into fixed size partition called page.

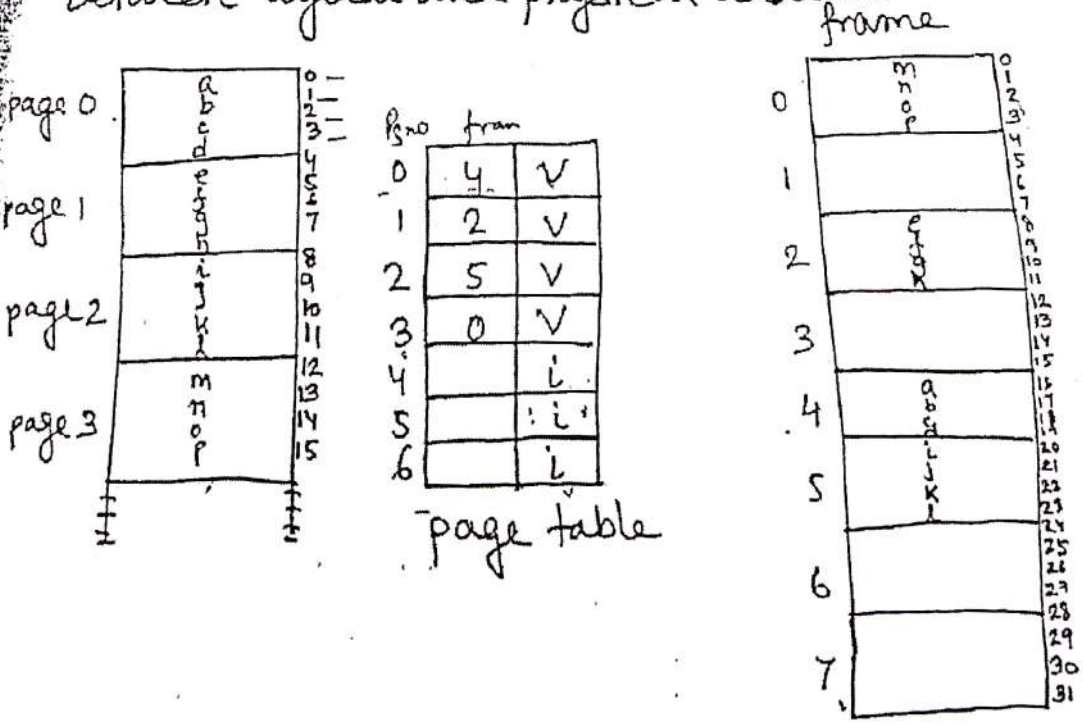
The size of the page and frame are equal. Logical address divided into page number (p) and offset (d). Page number is used to search the frame number from the page table. The following type of shows the paging hardware —



The following diagram shows the relationship between page and frame.



The following diagram shows the relationship between logical and physical address.



Find the physical address of k, c, n, h.

Physical address = frame no. x frame size + offset

$k = 5 \times 4 + 2$

$= 22$

$c = 4 \times 4 + 2$

$= 18$

$n = 0 \times 4 + 1$

$= 1$

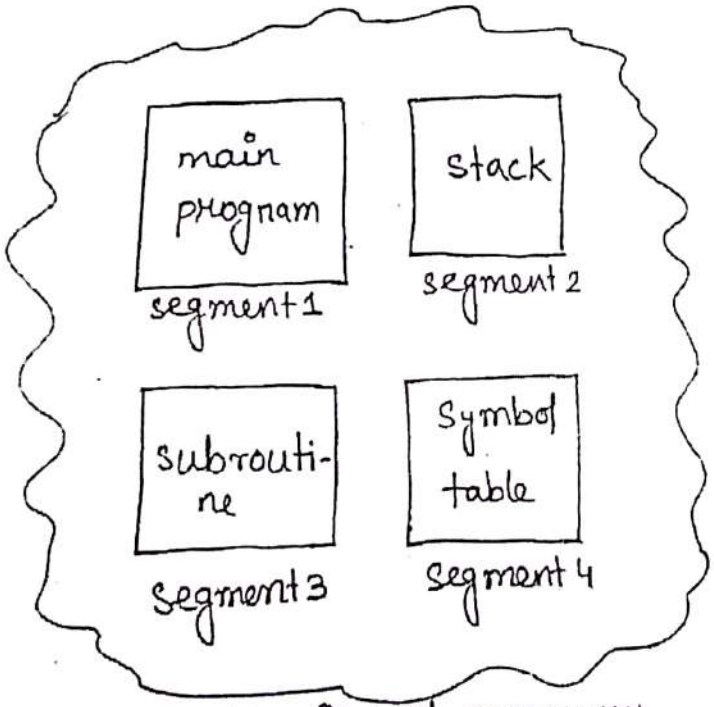
$h = 2 \times 4 + 3$

$= 11$

27.8.11

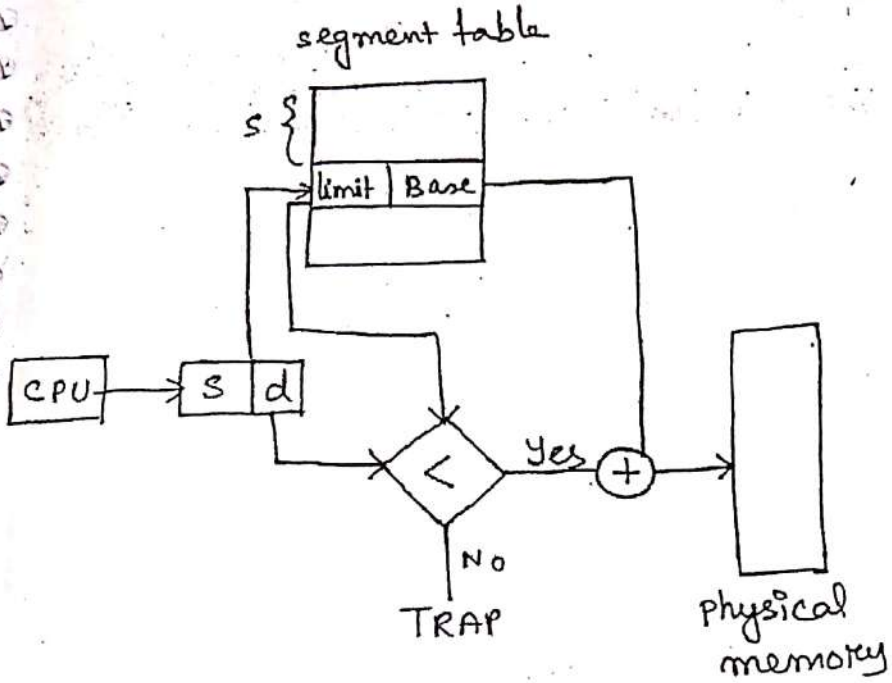
2f What is segmentation? Describe the technique of segmentation. Difference between paging and segmentation.

ans:- Segmentation is a memory management scheme that support user view of memory. A logical address space is a collection of segments. Each segment has a name and a length. The addresses specify ~~is~~ both the segment name and the offset within the segment. The following diagram shows the user view of a program.



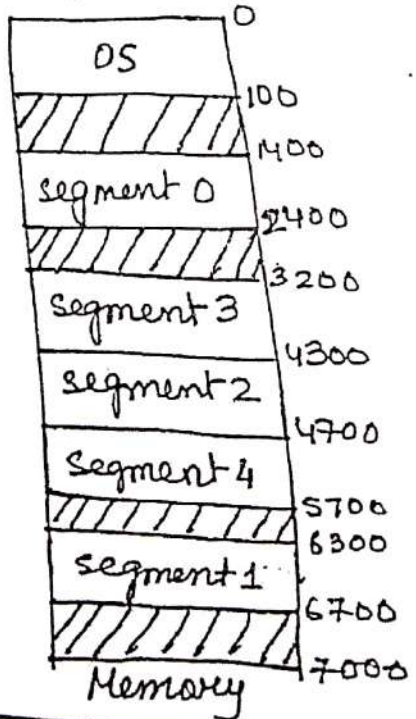
user view of memory

The segmentation hardware is given below



Each entry in the symbol table has a segment base and a segment limit. Segment base contains starting physical address whereas segment limit specify the length of the segment. For eg. Consider the following table

Segment no.	Base	Limit
0	1400	1000
1	6300	400
2	4300	400
3	3200	1100
4	4700	1000

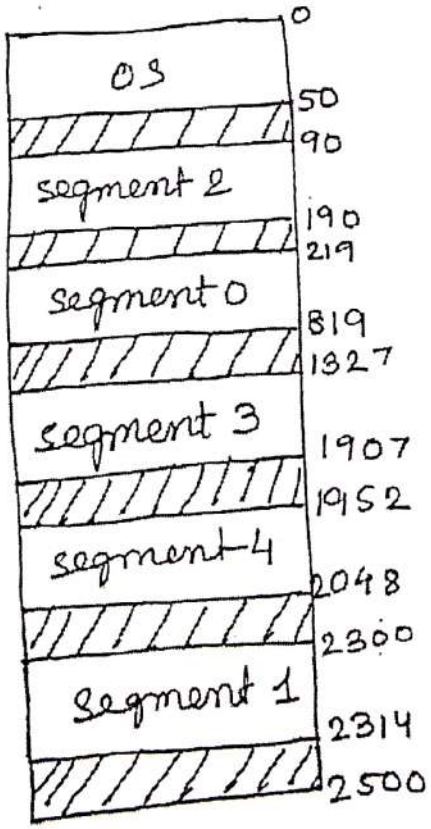


Consider the following segment table.

segment no.	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses

a) 0430
b) 110
c) 2500
d) 3400
e) 4112



a) 0430 (s, d) → 1400 + 430
= 1830

a) 0430 (s, d) → 219 + 430
= 649

b) 110 → 2300 + 10
= 2310

c) 2500 → No - Trap

d) 3400 → 1327 + 400
= 1727

e) 4112 → No - Trap.

array of storage. Separating logical memory as viewed by the user from ~~virtual~~ physical memory.

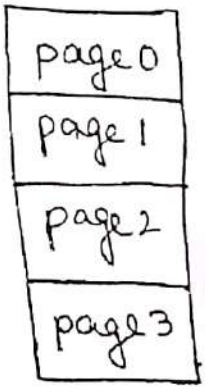
Virtual memory is the separation of user logical memory from physical memory. This separation allow an extremely large virtual memory to be programmed for the programmer. when only a smaller physical memory is available.

Virtual memory makes the task of programming much easier because the programmer no longer need to concern about the amount of physical memory.

Virtual memory can be implemented with the help of demand paging. A demand paging system is similar to paging system with swapping. Processes reside on secondary memory. when we want to execute a process we swap it into memory. Rather than swapping the entire process we use lazy swapper. A lazy swapper never swap a page into memory unless that page will be needed. Therefore

such kin

on dem
with se



Difference between paging and segmentation

Paging	Segmentation
i) Logical address is divided into page.	i) Logical address is divided into segment.
ii) Logical address has ~~segment no~~ page number (p) and offset (d).	iii) Logical address is divided into segment number (s) and offset (d).
iii) There is no ~~abstract~~ user view of memory.	iii) There is user view of memory.
iv) There is page table.	iv) There is segment tab
v) There are no base and limit register.	v) There are base and limit register.

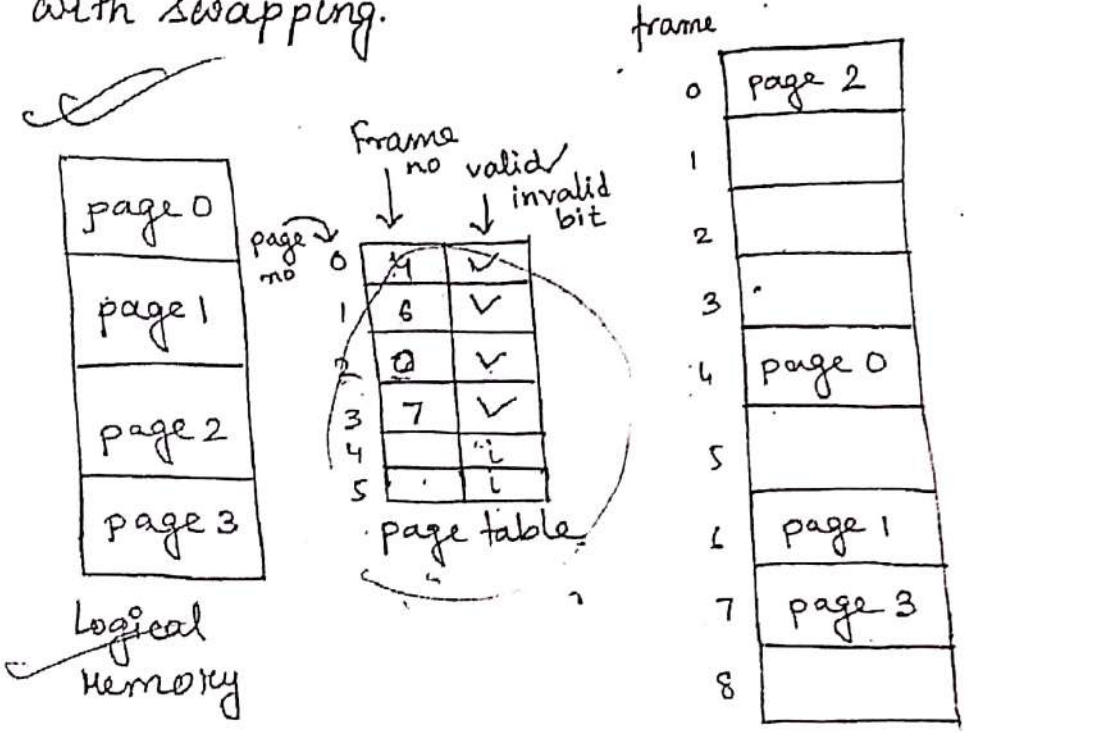
Virtual Memory and demand paging

Virtual memory is a technique that allow the execution of processes that may nd be completely in memory. One major advantag of this scheme is that programs can be large than physical memory.

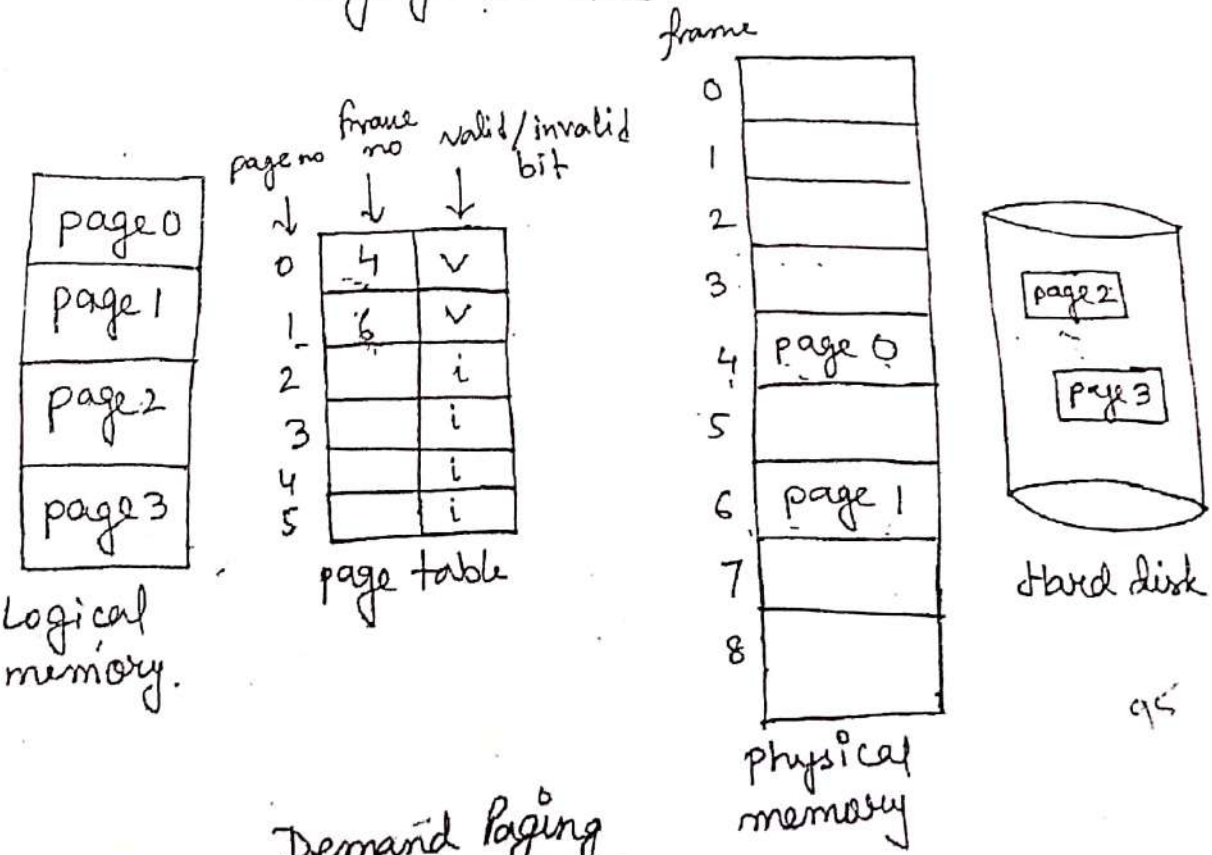
Virtual memory abstract main memory into an extremely large uniform

such kind of swapper is called pager.

Demand paging is basically paging on demand. It is a combination of paging with swapping.



Paging Technique



Demand Paging

What do you mean by page replacement? Describe different page replacement algorithm.

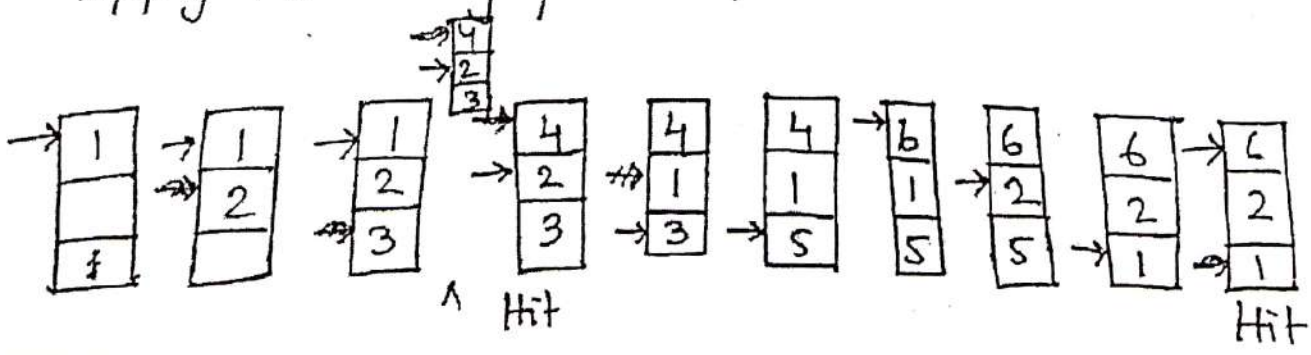
Page replacement is a technique of replacing the allocated frame with different page. There are three types of page replacement algorithm

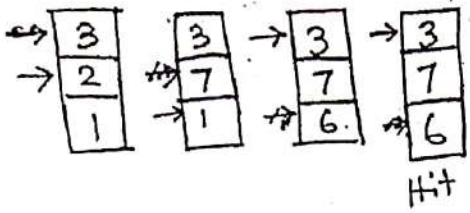
i) FIFO - It is one of the simplest page replacement algorithm.

b) FIFO page replacement algorithm associates with each page the time when the page was brought into memory. When a page must be replaced the oldest page is selected. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When a page is brought into memory we insert it at the tail of the queue.

For example:- Consider the following page reference string. 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3

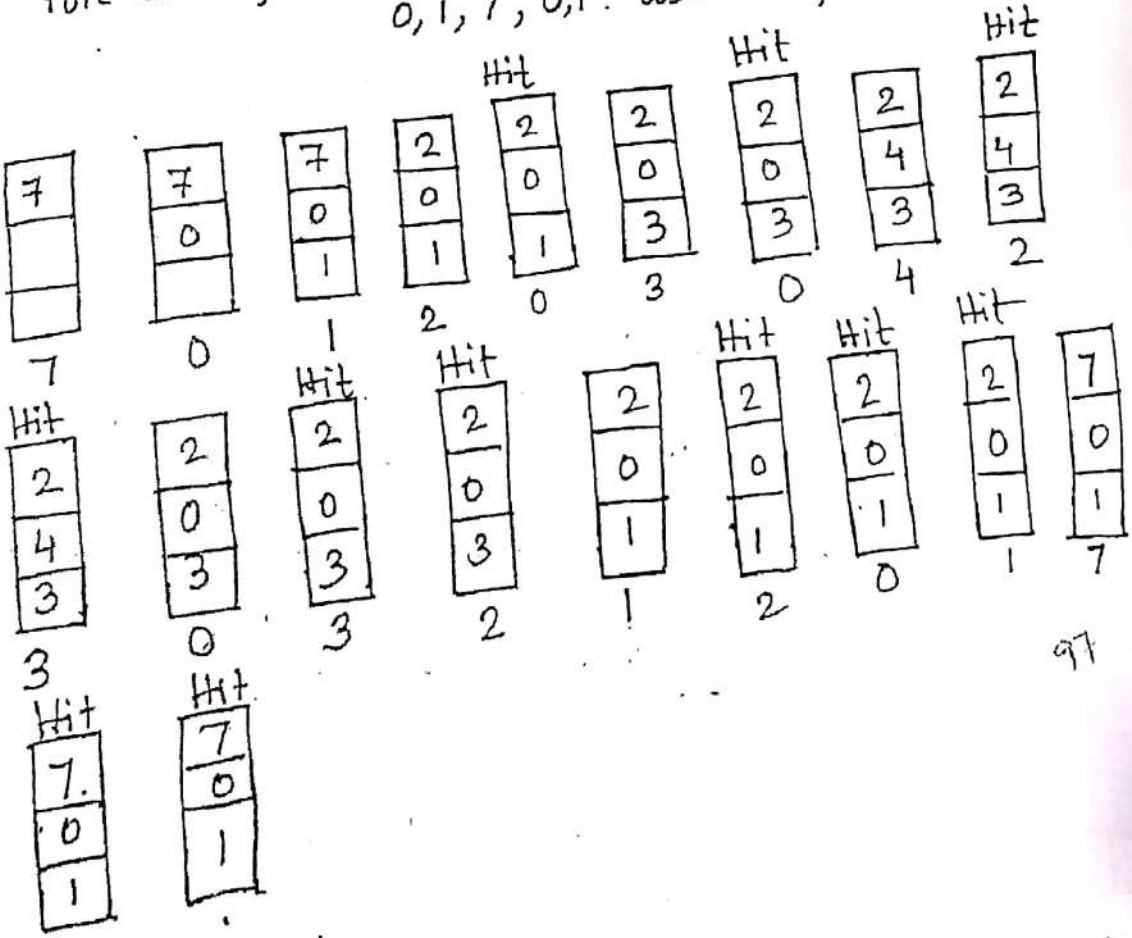
Apply FIFO using frame no. 3.





ii) Optimal page replacement algorithm has the lowest page fault rate of all algorithm. The criteria of this algorithm is "replace the page that will not be used for the longest period of time. It is a forward technique. The optimal page replacement algorithm is difficult to implement because it requires future knowledge of the reference string.

For example - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1. Use 3 frame

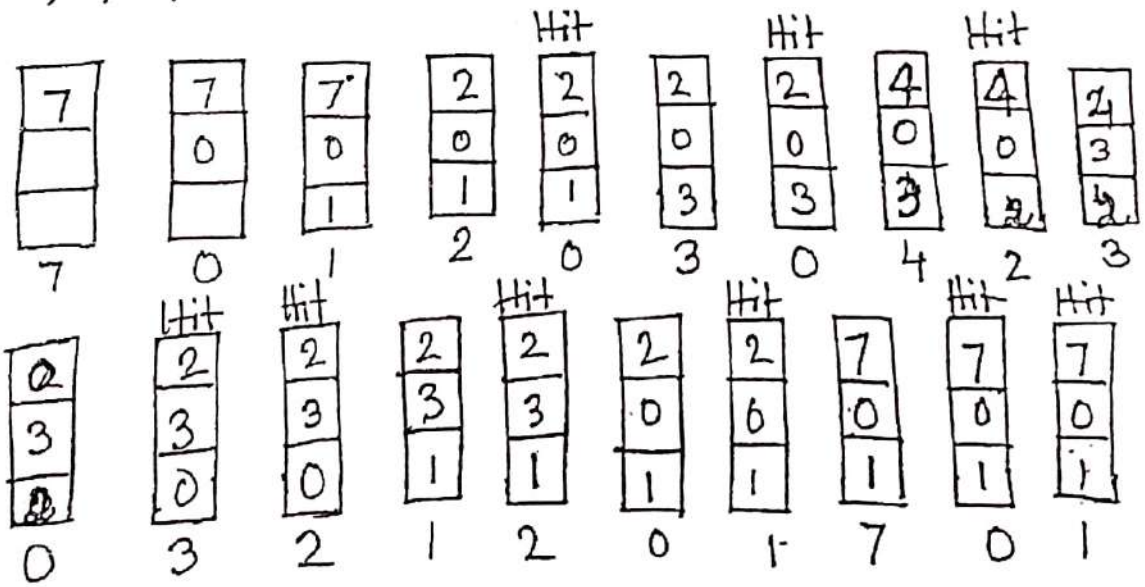


iii) LRU Page replacement algorithm - Full form of LRU page replacement algorithm is least recently used algorithm. This algorithm associate with each page the time of that page last use. When a page must be replaced LRU choose that page that has not been used for the longest period of time.

It is a backward technique. The main problem with LRU algorithm is the implementation overhead.

For example:-

Consider the reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1. Use 3 frame.



Consider the following page reference string ~~0,1,0,0,~~ 0100, 0432, 0101, 0612, 0102, 0103, 0104, 0101, 0611, 0102, 0103, 0104, 0101, 0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105. Calculate page fault rate for the following algorithm.

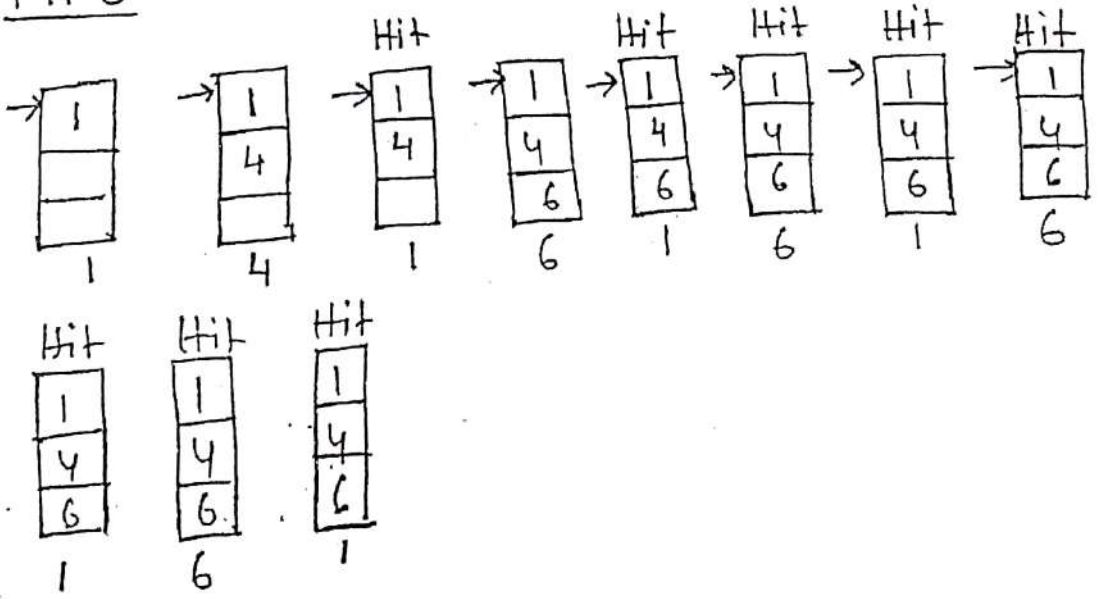
i) FIFO
ii) LRU
iii) Optimal

Memory size is 3 frame.

The page reference string is 1, 4, 1, 6, 1, 6, 1, 6, 1,

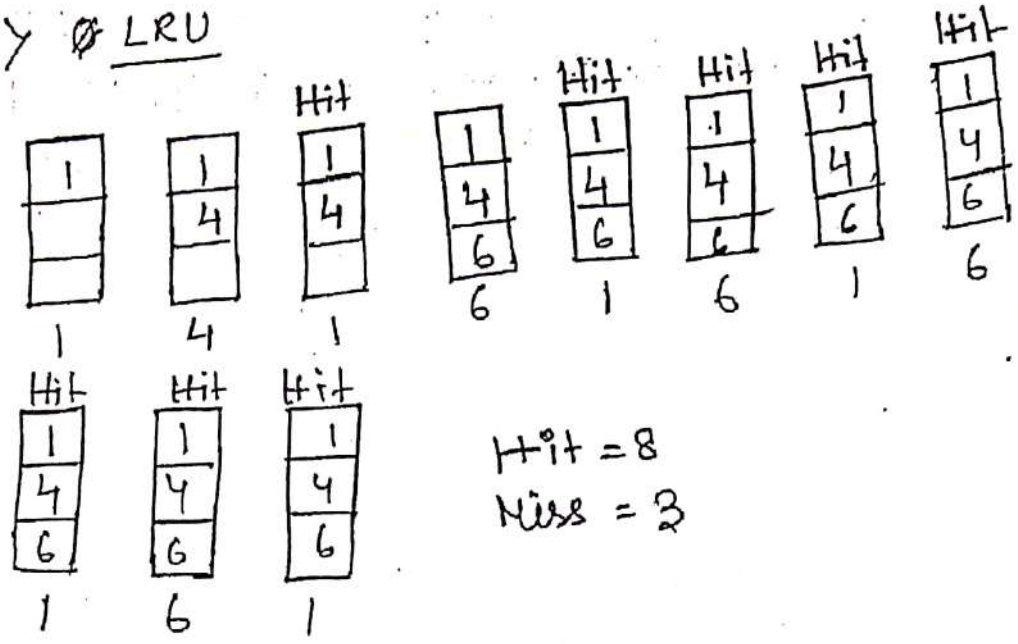
1.

i) FIFO



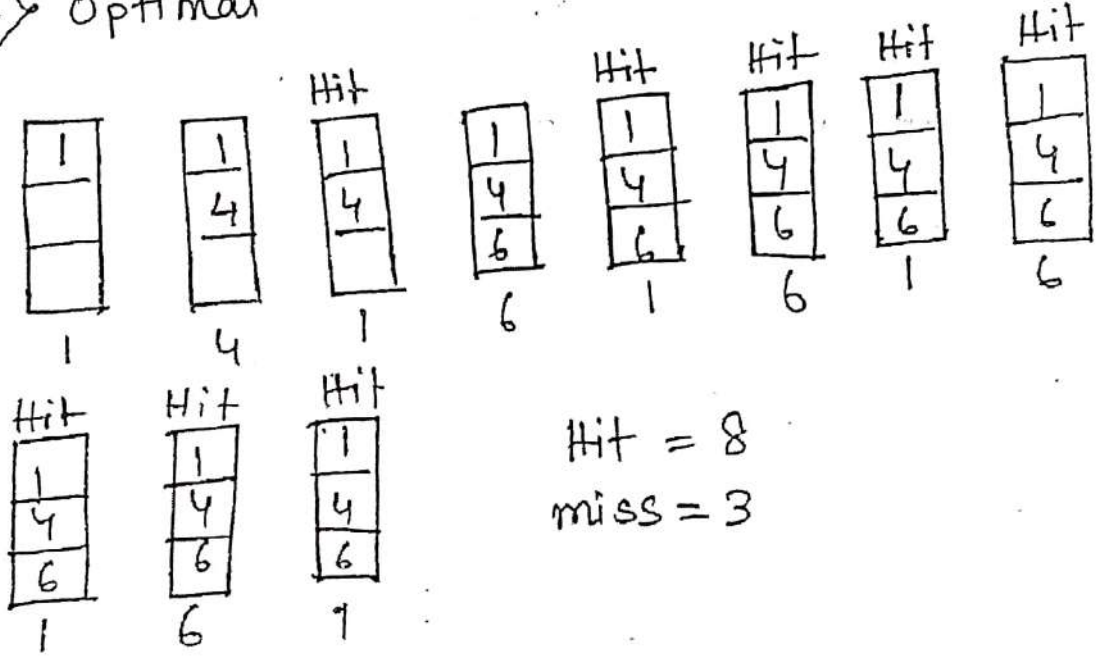
Hit = 8
Miss = 3.

ii) @ LRU



Hit = 8
Miss = 3

iii) Optimal



Hit = 8
miss = 3

6.9.11

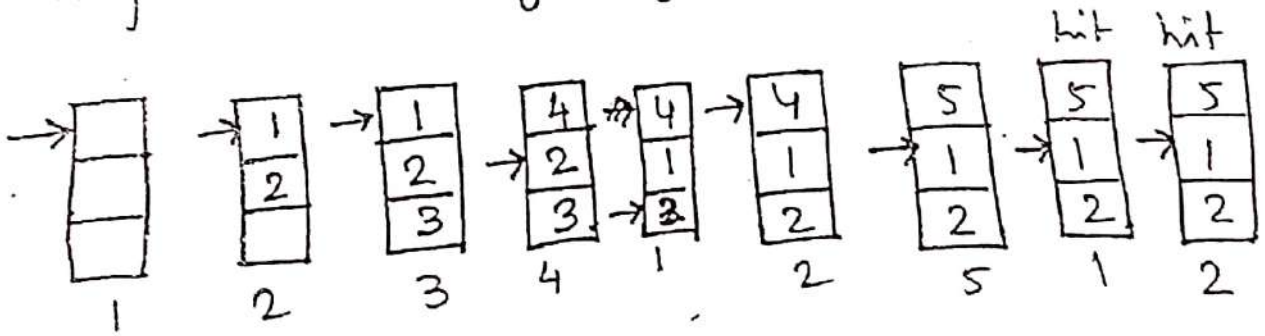
What is Belady Anamoly ?

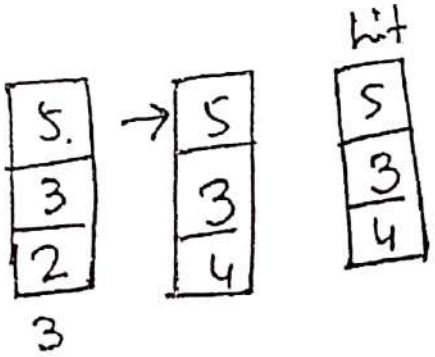
In FIFO page replacement algorithm if we increase number of allocated frame then page fault rate also increases this phenomenon is called Belady anamoly.

It is normal that if we increase the number of frame page fault rate much decreases but it is an unusual fact in FIFO page replacement algorithm. This is an unexpected result of increasing page fault rate inspite of increasing frame.

Consider the reference string.

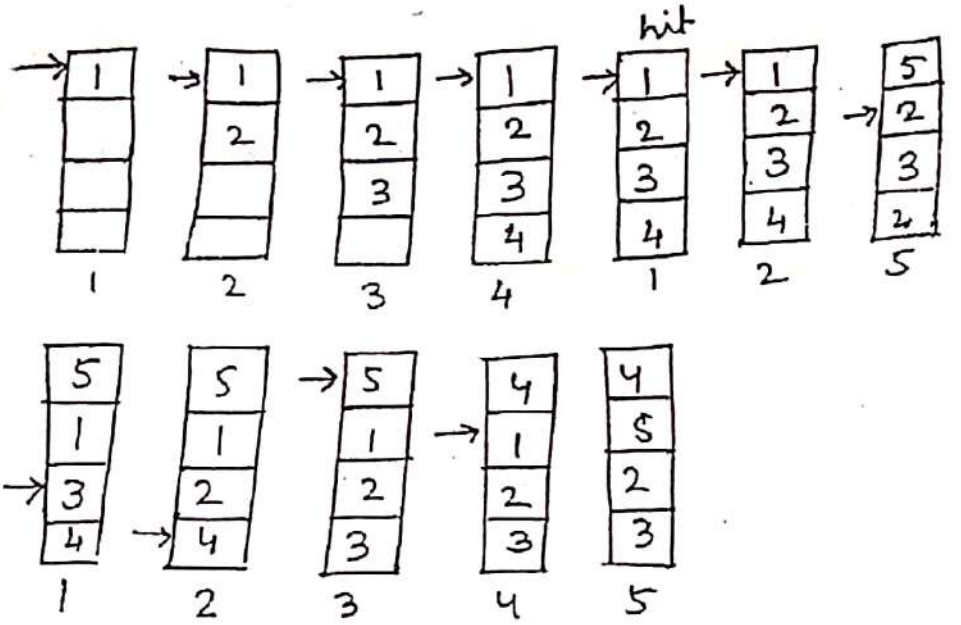
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5. Apply FIFO using 3 frame and four frame.





hit = 3
miss = 9

hit ratio = 3/12
miss ratio = 9/12
(page fault rate)



hit = 2 hit ratio = 2/12

miss = 10 miss ratio = 10/12

(page fault rate)

From the above example no. of misses increases though number of frame increases. This is called belady anamoly. This anamoly occur only in FIFO page replacement algorithm.

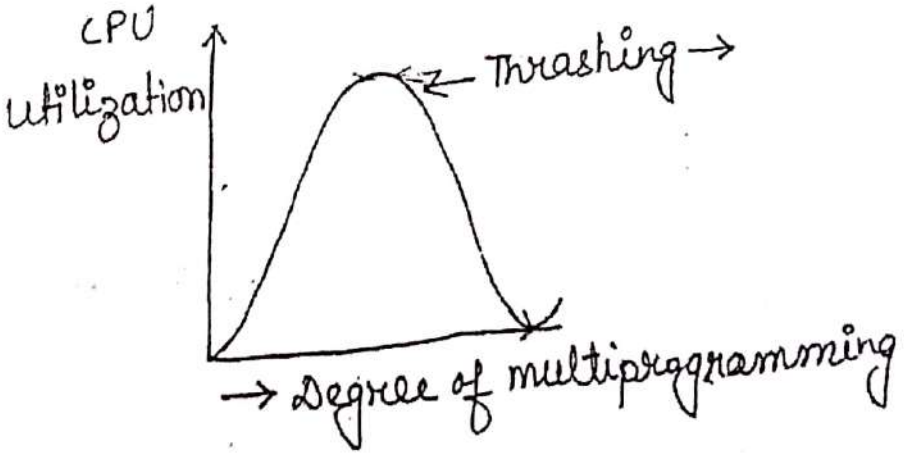
What is Thrashing? Describe with suitable diagram.

If the no. of frames allocated to a low priority process falls below the minimum number required by the computer architecture, we

must suspend that process execution. Thrashing result in different performance problem in computer system. The operating system monitors CPU utilization. If the CPU utilization is too low we increase the degree of multiprogramming by introducing new process in the system.

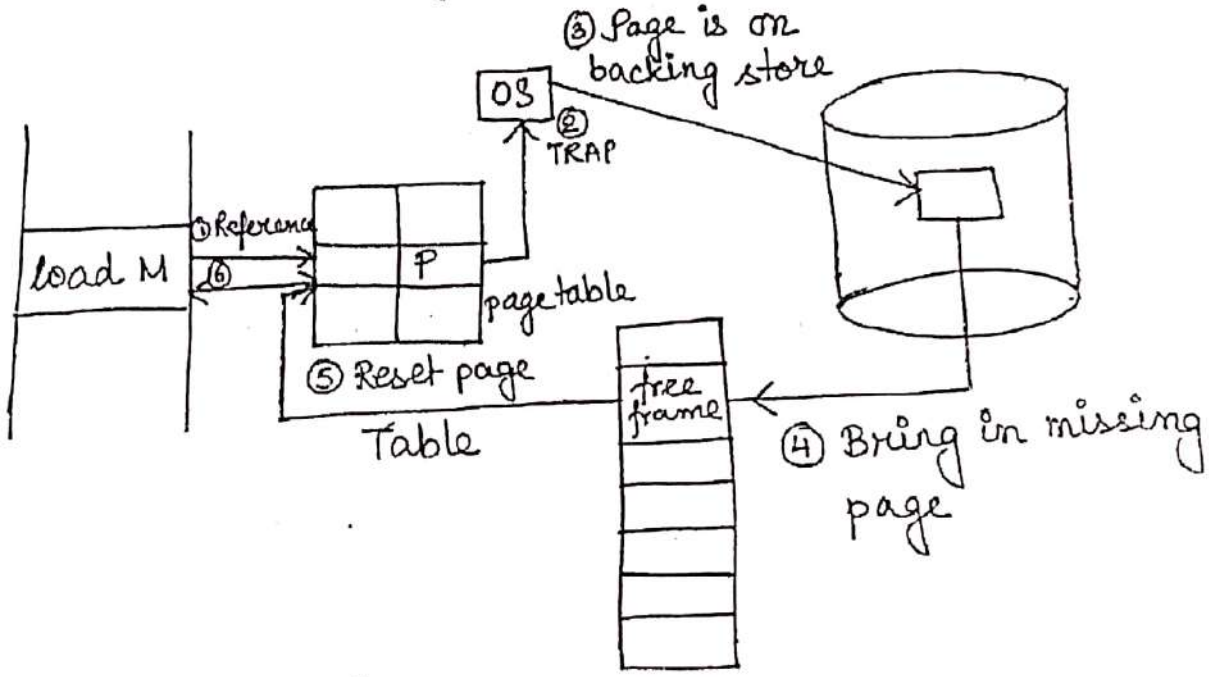
Now suppose that a process enter a new phase in its execution and needs more frames it starts faulting & and taking frames away from other processes. When the degree of multiprogramming increases CPU utilization also increases, although more slowly until a maximum is reached. If the degree of multiprogramming is increased even further, then CPU utilization drop sharply. This is called thrashing.

The following diagram shows the thrashing of operating system.



What is page fault? Describe the technique of page fault. (8)

When CPU refer to a page from the page table but it is not available then the page is need to bring from secondary memory to main memory, it is called page fault. The technique of page is given below :- (4) (5)



(6)

The steps are -

① CPU reference for a page from the page table. and the page is not available in the page table because there is invalid bit.

② The OS is called using Trap. Trap is one kind of interrupt. therefore control goes to OS.

③ The page is on the backing store (hard disk). The OS search the page from the hard disk.

④ After successful search for the page OS bring it into main memory and allocate the page to a free frame.

⑤ The free frame number is updated to the page table and it becomes valid.

⑥ Now the OS restart the instruction.

1) What is thread ? Difference between process and thread.

what is scheduling ? Describe long term, short term and ~~middle term~~ medium term scheduler.

Short note multilevel feedback queue scheduling

multilevel queue scheduling.

1) what is thread ? Difference between process and thread.

A thread is similar to a sequential program It has a beginning and end and a sequence. A thread itself is not a program because it cannot run on its own. So it runs within a program.

A thread is a single sequential flow of control within a program. A thread is also known as light weight process because it runs within a program and makes use of resources that are actually allocated to the process.

Multithreading - A process is divided into smaller task and each task is called a thread. The use of multiple thread in a single program all running at the same time and performing different task is called multithreading.

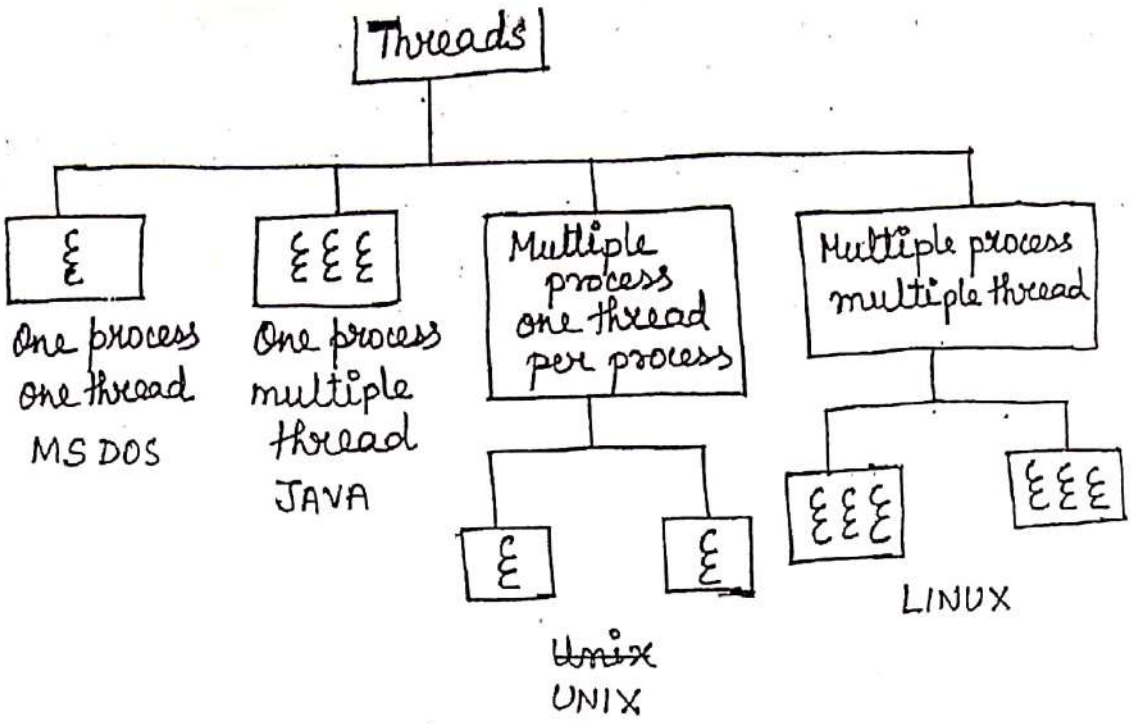
For example Java browser is a multithreaded application where we can scroll a page while it is downloading an image, play animation and sound parallely, print a page in the background while downloading a new page

There are four types of thread -

i) One process one thread
ii) One process multiple thread
iii) Multiple process one thread per process
iv) Multiple process multiple thread.

Following diagram shows different types of thread :-

P.T.O



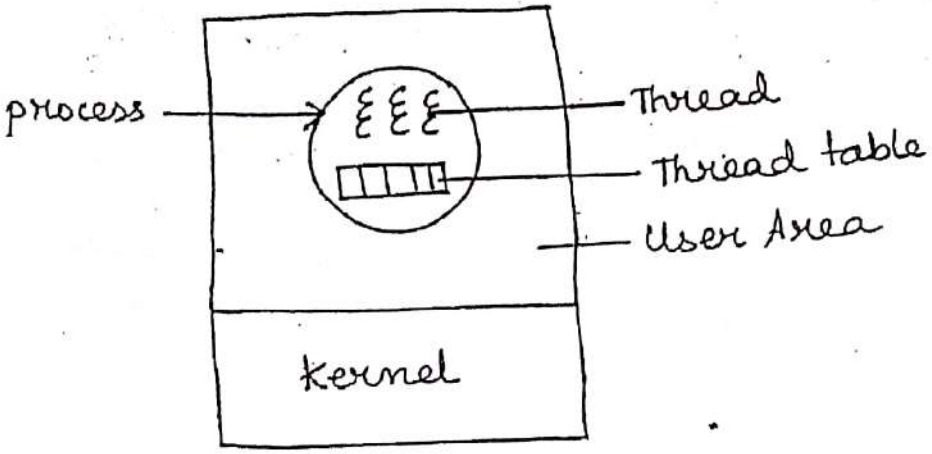
There are two ways of thread implementation -

i) User level thread

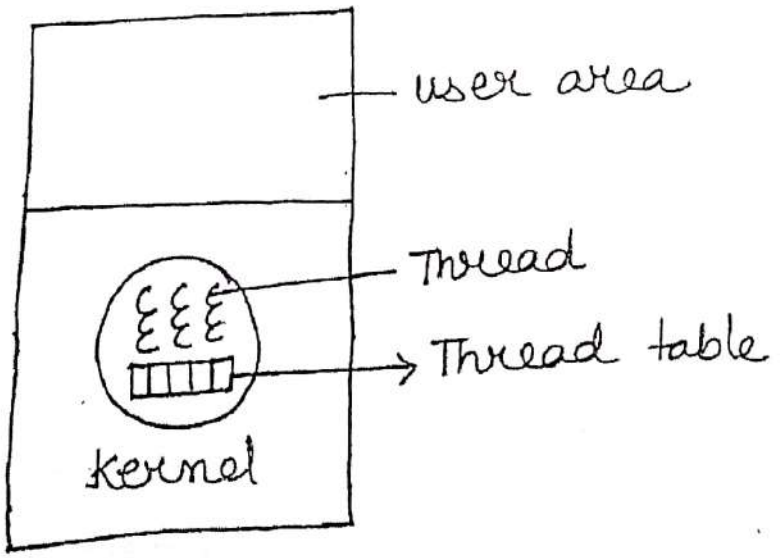
ii) Kernel level thread

i) <u>User level thread</u> - This type of threads are loaded in the users space only. So the kernel does not know anything about them. When threads are managed in the users space each process must have its own private thread cabin. The following diagram shows the user level thread:-

The thread consist of information about program counter, stack pointer, register etc and is managed by runtime system.



ii) Kernel level thread – In this method of thread implementation kernel does total work of thread movement. There is no thread table in each process. The kernel has a thread table. This table keeps track of all the thread in the system. Using kernel call only a thread can create new thread or destroy an existing thread.



Difference between process and thread

Process	Thread
i) Processes cannot share the same memory.	i) Thread can share memory and files.
ii) Process creation is time consuming.	ii) Thread creation is not time consuming.
iii) Process execution is very slow.	iii) Thread execution is very fast.
iv) It takes more time to terminate the process.	iv) It takes less time to terminate threads.
v) Process is loosely coupled.	v) Threads are tightly coupled.
vi) System calls are required for communication.	vi) System calls are not required.
vii) Communication between process is difficult.	vii) Communication between thread is easy and efficient.

Year 2006

Group-A (MCQs)

i) Virtual memory is
a) an extremely large main memory
b) extremely large secondary memory
c) an illusion of extremely large memory.
d) a type of memory used in super computer.

ii) Page fault occur when
a) the page is corrected by application software
b) the page is in main memory
c) the page is not in main memory
d) one tries to divide a number by 0

iii) Concurrent processes are process in
a) do not overlap time
b) overlap in time
c) are executed by process at the same time
d) none of these.

iv) Mutual exclusion problem occurs between
a) two adjacent processes that do not interact.
b) processes that share resources

c) processes that do not use the same resources.

d) none of these

v) To avoid race condition the maximum number of processes that may be simultaneously inside the critical section

a) 100

b) 1

c) 2

d) 3

vi) When does an interrupt occur the operating system ?

a) Ignore the interrupt

b) Always change ~~step~~ state of interrupted process after processing the interrupt.

c) Always resume execution of interrupted process after processing the interrupt.

d) May change state of interrupted process to block and schedule another process.

vii) which is not a layer of operating system ?

a) Kernel

b) shell
c) application program
d) critical section

viii) With a single resource deadlock occurs
a) if there are more than two processes competing for that resource.
b) if there are only two processes competing for that resource.
c) If there is a single process competing for that resource
d) none of these

ix) Thrashing
a) reduce page I/O
b) decreases the degree of multiprogramming
c) implies excessive page I/O
d) impresses the system performance.

x) In which of the following scheduling policies does context switching never take place?
a) RR
b) SJF
c) Preemptive
d) FCFS

Year 2007

Group - A

i) Which is not a layer of OS?

Refer to <1vi> of 2006

ii) Thrashing

a) always occurs on large computers

b) is a natural consequence of virtual memory system

c) Can always be avoided by swapping.

d) can be call by 4 paging algorithm

iii) PCB stand for

a) Programming Control Block

b) Parallel Control Block

c) ~~Par~~ Process Control Block

d) none of these

iv) RMI stands for

a) Remote method interface

b) Remote message interface

c) Remote method invocation

d) none of these

v) Necessary condition for deadlock is
a) Mutual exclusion
b) Hold and wait
c) No preemption
d) all of these

vi) Which is preemptive scheduling?
a) FCFS
b) SJF
c) Both (a) and (b)
d) None of these

vii) Which is not a valid process state?
a) new
b) ready
c) run
d) load

viii) Which one of the following is not page replacement algorithm?
a) LRU
b) Memory replacement
c) Process replacement
d) Optimal

ix) Shift time related to
a) CPU scheduling
b) disk "
c) Page fault
d) memory management

Year 2000

Group-A (MCQ)

i) Banker algorithm is used for *
a) mutual exclusion
b) paging
c) deadlock detection
d) none of these

ii) which is non preemptive scheduling algorithm?
a) RR
b) FCFS
c) Priority
d) None of these

iii) which kind of scheduler are responsible for process suspension and activation?
a) short term.
b) medium term
c) long term
d) both (b) and (c)

iv) Semaphore is a
a) function
b) variable
c) macro
d) none of these

116

20.9.11

Short note on worm and virus

Most operating system provide a means for processes. A spawn other processes in search an environment, it is possible to create a situation where operating system resources and user files are unused. The two most common method for achieving this misuse are worm and virus.

Worm – A worm is a process that uses a spawn mechanism to hamper the system performance. The worm spawn copies of itself using system resources and locking out system used by all other processes. The worm was made up of two programs –

1) Grappling hook – also called bootstrap program.
2) Main program

Once established on the computer system under attack the grappling hook connected to the machine where it originated and uploaded a copy of the main worm on to the hook system.

The main program proceed to search other machine to which the newly infected system would connect easily.

Virus - Another form of computer attack is virus. (Vital Information resources under sketch. Like worm virus are designed to spread other programs and can modify the destroyed files causing system.

A worm is a structure stand alone program. Virus is a fragment of code. Virus are a major problem for computer users specially micro computer. Virus are generally spreads by users downloading internet program. Antivirus is very useful to prevent the attack of virus and save the computer resources.

Antivirus needs to be regularly scanned.

Cryptography

Cryptography is one of the technique of providing data security during transmission

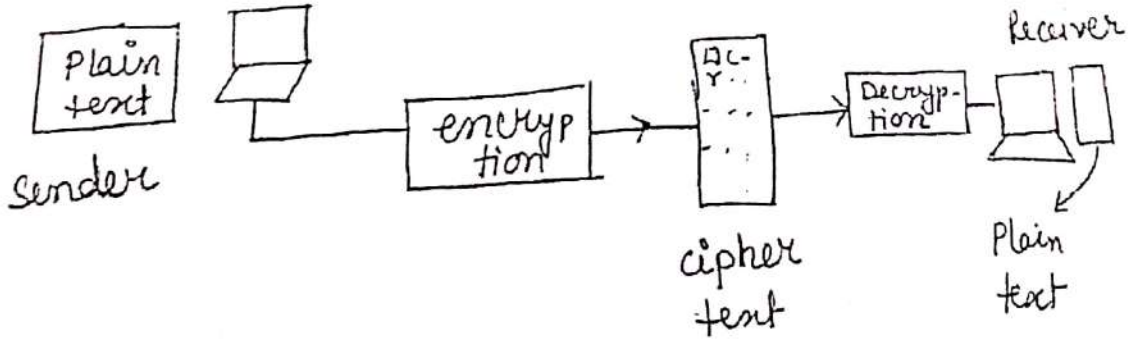
of data from one machine to another machine. It is a set of methods through which we can deliver messages from one machine to another machine in a secure way. A cryptography consist of two algos -

1) Encryption algorithm
2) Decryption algorithm

Encryption algo convert the plain text into cipher text.

Decryption algo convert cipher text into plain text.

There are some encryption and decryption key to convert the plain text to cipher text and vice versa.



11:4

23.4.11

1) Describe different allocation method.

The direct access nature of disc allow us flexibility in the implementation of files. In almost every case many files will be stored on the same disc. The main problem is how to allocate space to these files so that the disk space is utilized and the files can be accessed quickly. There are three methods of allocating disk space -

i) contiguous allocation
ii) linked allocation
iii) indexed allocation

i) Contiguous allocation - This allocation method require each file to occupy a set of contiguous block on the hard disk. The disk addresses define linear ordering in the disk.

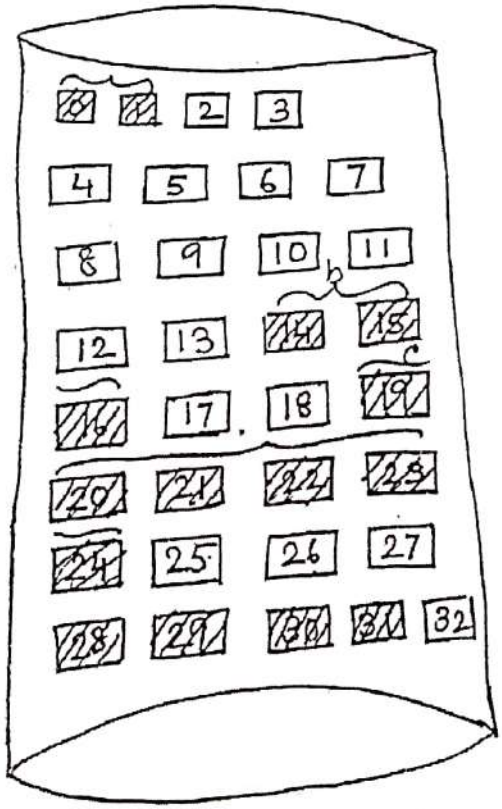
Contiguous allocation of files is defined by the disk address and the length of the first block. If the file is n block long and start at location b then it occupies

block $b, b+1, b+2, \ldots b+n-1$

Consider the following diagram.

Directory table

File name	Start	Length
A	0	2
B	14	3
C	19	36
D	28	64
E	6	42

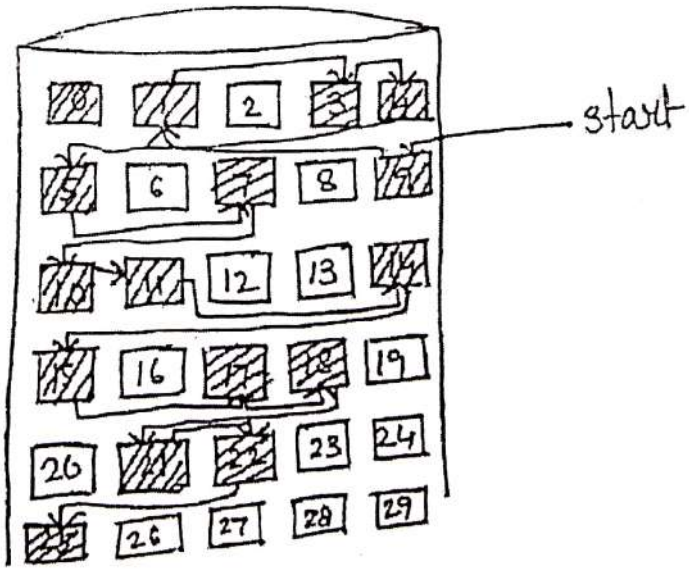


This algorithm suffers from the problem of external fragmentation. Another problem is

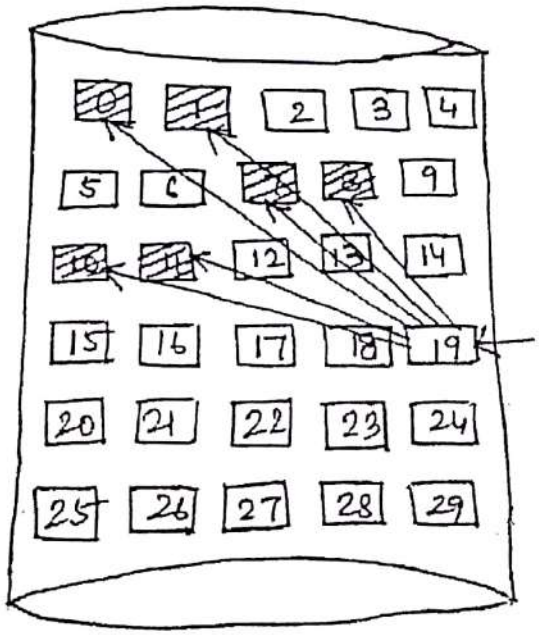
determining how much space is needed for a file. When the file is created the total amount of space it will need must be found and allocated.

ii) Linked allocation - The linked allocation solve all problems of contiguous allocation. With linked allocation each file is a linked list of disk block. The disk block may be scattered anywhere on the disk. The directory contains a pointer to the first and last block. To create a new file we simply create new entry in the directory. Consider an example: Consider the table.

File	Start	End
A	9	25



iii) Indexed allocation - Linked allocation solve the external fragmentation and size declaration problem of contiguous allocation. However in the absence of file allocation table the linked allocation cannot support efficient direct access. Since the pointer to the block are scattered. Indexed allocation solve this problem while bringing the pointer together into one location which is called indexed block. Consider an example.



Directory table

File name	Index block
A	19

Year - 2006

Group B

2) Explain the difference between external and internal fragmentation.
Refer to the definition and eg. of first fit

3) What is semaphore? Explain its significance
Def., Implementation

4) Threads are the convenient mechanism for concurrency control within the application - Explain.
Refer to thread.

5) When do page fault occur? Describe the action taken by os when page fault occur.
Refer to page fault.

6) Explain the role of virtual memory in OS.
Refer to virtual memory & demand paging.

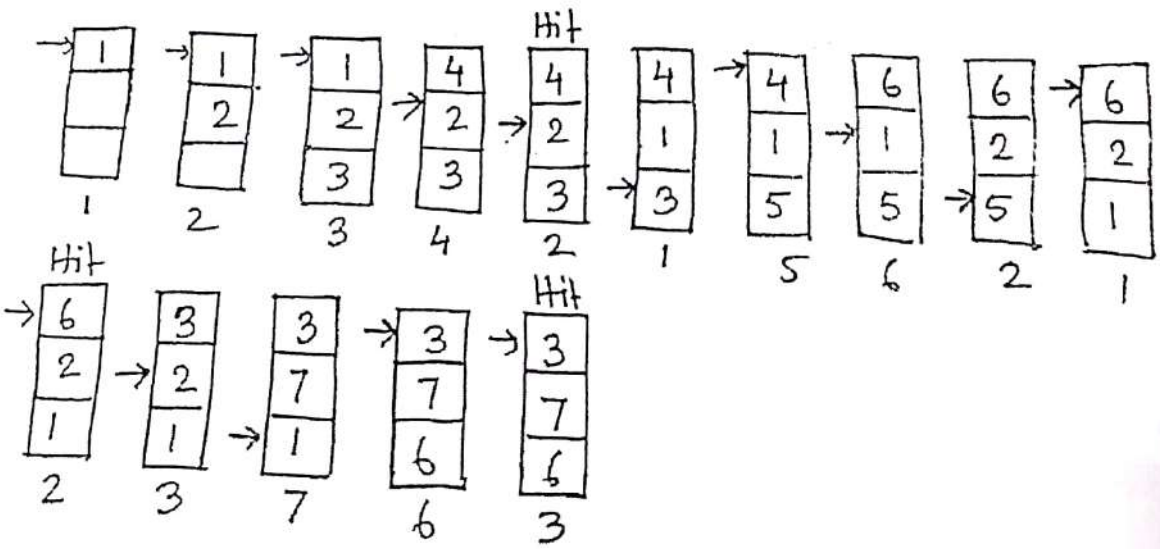
Group-C

7) Explain the following :-

a) Belady anamoly
b) Compaction
c) Process control Block

} Refer to note

8) Consider the following page reference stream 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3. How many page fault would occur for FIFO replacement algorithm using frame no. 3.



2)

Short note on multiprocessor operating system

Most system are single processor system. They have only one main CPU. However multiprocessor system are growing in importance. Such system have more than one processor in closed communication. Sharing the computer bus the clock and sometimes the memory and peripheral devices.

Multiprocessor system has three main advantages –

i) Increased throughput

ii) Economy of scale

iii) Increased reliability

There are two types of multiprocessor system –

) Symmetric multiprocessing where each processor runs an identical copy of the operating system. and this copies communicate

with one another.

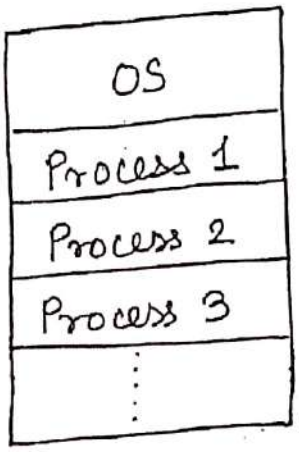
2) Asymmetric multiprocessing in which each processor is assigned a specific task. A master processor controls the system and other processor either look to the master for instruction or have predefined task. There is a master slave relationship. The master processor schedules and allocate work to the slave processor.

Multiprogramming System

The most important aspect of job scheduling is the ability to multiprogram. A single user cannot keep the CPU or the I/O devices busy at all times. Multiprogramming increases CPU utilization by organizing jobs so that CPU always has one to execute.

Idea is as simple. The Operating System gives several jobs in memory simultaneously. The set of jobs is a subset of jobs kept in a job pool. Since the number of jobs that can be kept simultaneously in memory is generally much smaller than the number

of jobs that can be in the job pool. The OS picks and begin to execute one of the jobs in the memory. and the CPU never sits idle.



Multitasking Operating System

Multitasking or time sharing is a logical extension of multiprogramming. CPU execute multiple jobs by switching among them but the switching occurs so frequently so that the user can interact with each program while it is running.

A multitasking operating system allow many user to share the computer simultaneously. Since each action and command

in a time shared system to be short therefore only a little CPU time is needed.

Distributed Operating System

A distributed system is a collection of all processors that do not share memory or clock. Instead its processor has its own local memory, the processor communicate with one another through various communication tools such as high speed buses and telephone lines. A distributed system is a collection of loosely coupled processors interconnected by communication network. From the point of view of a specific processor in a distributed system the rest of processor and their respective resources are remote. whereas its own resources are local. Advantages of distributed system are —

i) Resource sharing
ii) Computation speed up
iii) Reliability
iv) Communication

There are various types of distributed OS:-

i) Network Operating System
ii) Distributed Operating System

ibuted

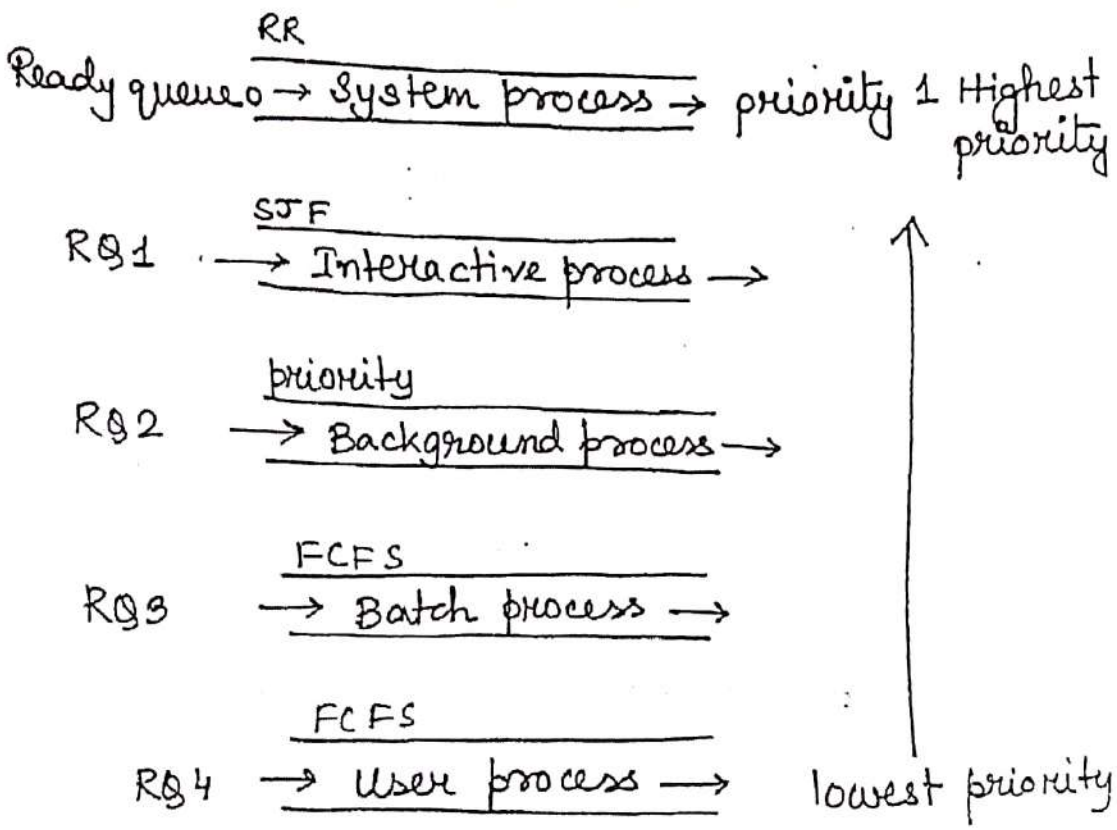
Multilevel queue scheduling

Another class of scheduling algorithm has been created for situation in which processes are easily classified into different groups. A multilevel queue scheduling algorithm partition the ready queue into several separate queues. The processes are permanently assigned to one queue generally based on some property of the process such as memory size, process priority or process time.

Each queue has its own scheduling algorithm. Consider an example. Suppose there are 5 processes. Therefore there are 5 types of queue.

i) system processes
ii) interactive processes
iii) background processes
iv) batch processes
v) user processes

The diagram of multilevel queue scheduling is given below:-



Each ready queue has different scheduling algorithm. No process execute until all the processes above the current ready queue is empty.

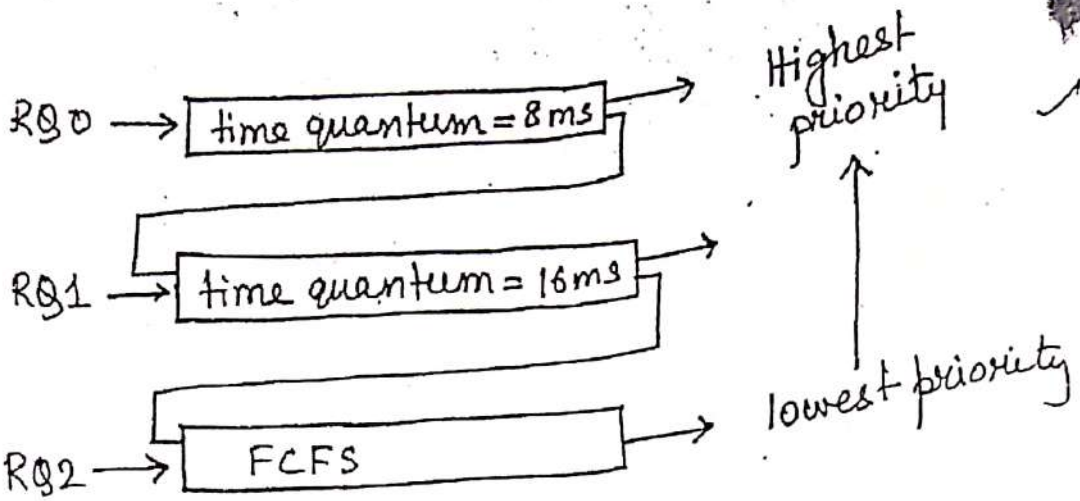
Multilevel feedback queue scheduling algorithm

Normally in a multilevel queue scheduling algorithm processes are permanently assigned

eduling to a queue on entry to the system. Processes do not move between queue. If there are separate queue then this can be a disadvantage

In multilevel feedback queue scheduling a process is allowed to move between queue. The idea is to separate processes with different CPU burst characteristic. If a process use too much CPU time it will be moved to lower priority queue. Similarly, a process that ~~weights too~~ waits too long in lower priority queue may be moved to higher priority queue.

Consider a multilevel feedback queue scheduler with 3 queues numbered from 0 to 2: The scheduler first execute all processes in queue 0. Only when queue 0 is empty it will execute processes in queue 1. Consider the diagram.



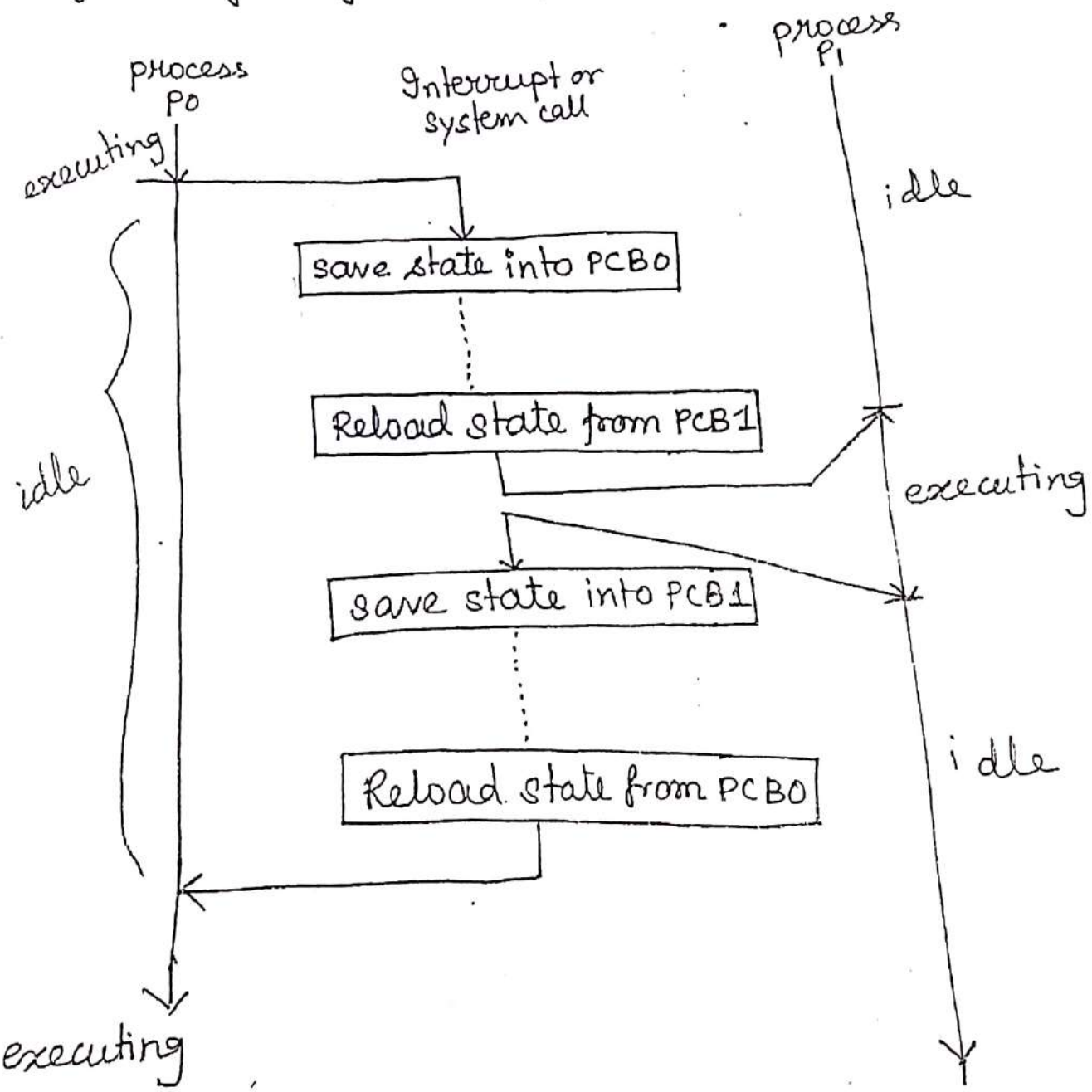
What is context switching?

Switching the CPU to another process require saving the state of old process and loading the ~~same~~ saved state for the new process. This task is known as context switch.

The context of a process is represented in the PCB of a process. When a context switch occur the kernel save the context of the old process in its PCB and load the saved context of the new process scheduled to run. The general speed of context switching range from

1 to 1000 micro seconds.

Context switching time are highly dependent on hardware support. A context switch simply includes changing the pointer to the current register set. Consider the following diagram.



Difference between deadlock and starvation.

The implementation of semaphore with a waiting queue may result in a situation where two or more processes are waiting indifferently for an event that can be caused only by one of the waiting processes. The event in question is the execution of the signal operation. When such a state is reached this processes are said to be deadlock.

Eg:- Consider a system where two processes P_0 and P_1 using two semaphore s and Q

P_0

```
wait (s);
wait (Q);
  :
  :
signal (s);
signal (Q);
```

P_1

```
wait (Q);
wait (s);
  :
  :
signal (Q);
signal (s);
```

Suppose, P_0 execute wait (s) then P_1 execute wait (Q). When P_0 execute wait (Q)

it must wait until P_1 execute signal(S). Similarly P_1 execute wait(S), it must wait until P_0 execute signal(S). Since, this signal operation cannot be executed, therefore P_0 and P_1 is deadlock.

Another problem related to deadlock is indefinite blocking or starvation, a situation where processes wait indifferently within the semaphore. Indefinite blocking may occur if we add or remove processes from the list associated with a semaphore in LIFO order.

Short note on system call

System call provides the interface between a process and the operating system. This calls are generally available as assembly language instruction and they are generally listed in various manual used by assembly language programmers. Certain system allow system calls to be made directly from a high level program in which case calls normally predefined function or subroutine. They may

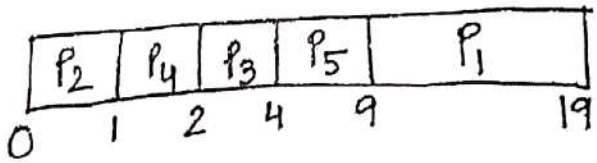
generate a call to a special runtime routine that make the system call or the system call may be generated directly.
C, C++ is used to make system call.
Some of the examples of system call are read, write, open, close. etc.

Consider a set of processes

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

Apply FCFS, SJF, Priority, RR(1)

Gnatt chart



Waiting time calculation :

Waiting time for $P_2 = 0$

$P_4 = 1$

$P_3 = 2$

$P_5 = 4$

$P_1 = 9$

Average waiting time $= \frac{16}{5} = 3.2$

Turn around time for

4.11.11

Year – 2009

MCQ

GR-A

i) Long term scheduler is also known as –

a) admission scheduler

b) dispatch "

c) swapping "

d) process "

e) none of these

ii) To avoid race condition number of processes that may be simultaneously inside the critical section is –

a) 0 c) 2 e) 5

b) 1 d) 4

iii) Thrashing is

a) reduce page I/O

b) implies excessive page I/O

c) decrease the degree of multiprogramming

d) improve the system information

e) none of these

iv) Interprocess communication ~~is never necessary~~
a) is never necessary
b) allow process to synchronize activity

b)

c) is required for all process
d) is usually done via drives
e) None of these

v) With a segment if there are 64 segment & maximum size is 512 word. The length of the logical address is -

a) 12 c) 15 e) 10
b) 14 d) 16

$512 = 2^8 \quad 64 = 2^6$

$\therefore L.A = 6 + 8$
$= 14$

vi) Operating System is responsible for
a) controlling peripheral devices such as monitor, printer, disc.
b) detecting errors in user programs.
c) providing an interface that allow user to choose program to run and manipulate files.
d) All of these

vii) When an interrupt occurs the operating system

a) ignore the interrupt

b) always change state of interrupted process after processing the interrupt.

c) always resume execution of interrupted processes after processing the interrupt

d) schedule another process

viii) Context switching is -

a) Part of spooling

b) Part of poling

c) Part of interrupt handling

d) Part of interrupt service

ix) Fork() is .

a) Creation of new job

b) Termination of a job

c) Increment of Task priority

d) None of these

x) Producer Consumer problem solved by
a) Semaphore
b) Event counter
c) monitor
d) All of these

Year - 2010

MCQ

GR-A

i) What is shell?
a) It is a hardware component
b) It is a command interpreter
c) It is a part of compiler
d) It is a tool of CPU scheduling

ii) Virtual memory is
a) An extremely large memory
b) An extremely large secondary memory
c) An illusion of extremely large storage provision
d) A type of memory used in super computer

iii) Multiprogramming system
a) are easier to develop than single programming system
b) execute each job faster

c) execute more job at the same time
d) are used only on large mainframe computers.

iv) which is not the state of process
a) Blocked
b) Running
c) Ready
d) Priviledged

v) The number of process completed per unit time is known as
a) output
b) Throughput
c) Efficiency
d) Capacity

vi) A critical region
a) a piece of code execute only one process at a time
b) is a region to deadlock
c) is a piece of code which execute only a finite number of process

d) is found only in windows operating system.

vii) The mechanism that bring page to memory only when it is needed

a) Segmentation
b) Fragmentation
c) Demand paging
d) Page replacement

viii) → c) b)

ix) → b) d)

x) FIFO scheduling is

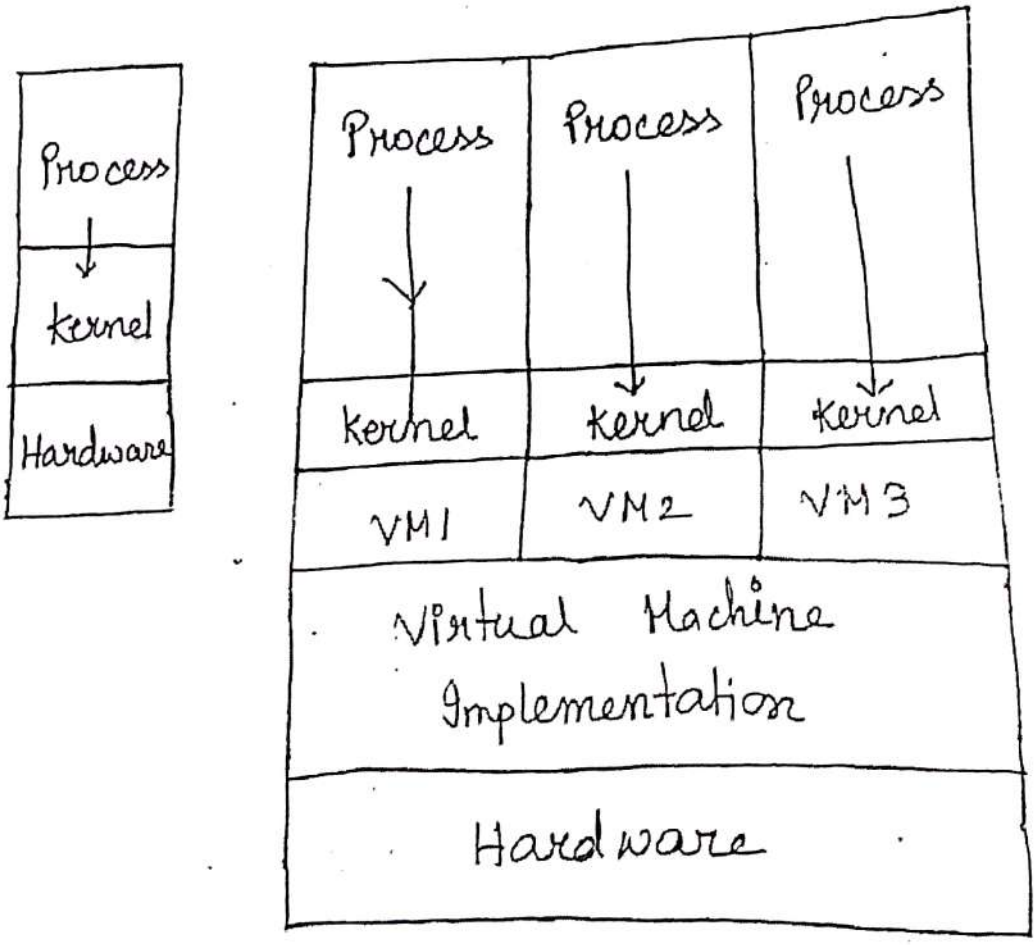
a) Preemptive scheduling
b) Non preemptive "
c) Deadline "
d) Fair share scheduling

Short note on Virtual machine

Although the system program are at level higher than that of the other routines, the application program may view everything under them in the hierarchy as though the latter were part of the machine itself. This a

layered approach is taken to its logical conclusion in the concept of virtual machine

The physical computer share resources to create the virtual machine. CPU scheduling can share out the CPU to create the appearance that user have their own processor. The following diagram shows how a non virtual and virtual machine operate.

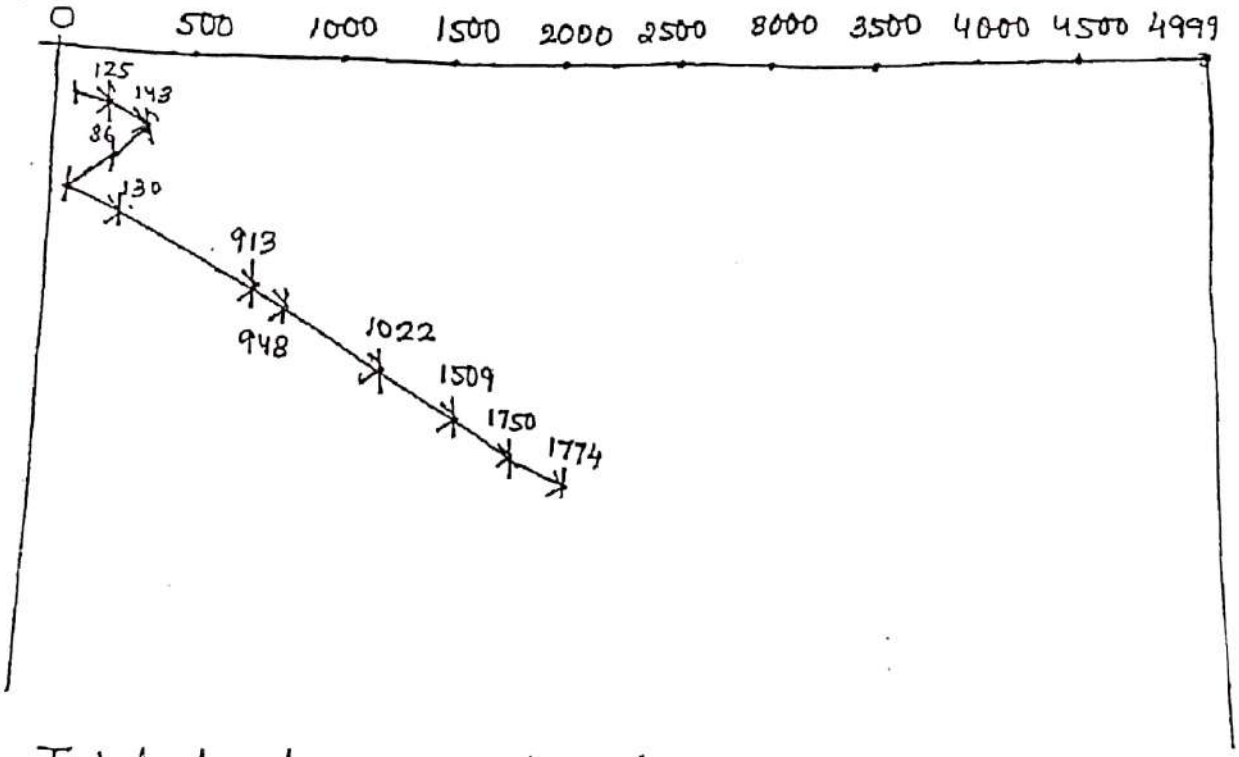


1.11.11

2006

Suppose that a disk drive has 5000 cylinder no. from 0-4999, the drive is currently serving a request at 143 previous request was 125. The queue of pending request is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. Starting from the current head position what is the total distance covered by the following disk scheduling algo- SSTF & SCAN

SSTF



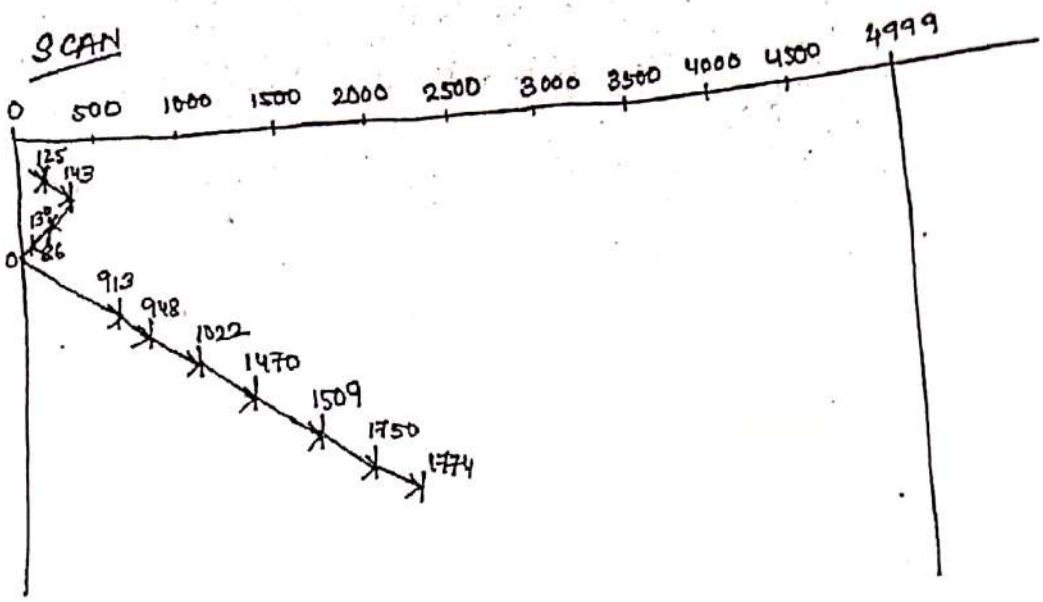
Total head movement = (143-130) + (130-86) + (913-86) + (948-913) + (1022-948) + (1470-1022) + (1509-1470) + (1750-1509) + (1774-1750)

= 13 + 44 + 827 + 35 + 74 + 448 + 39 + 241 + 24

= 1745

Average = 193.8

SCAN



Total head movement

$(143-130)+(130-86)+(86-0)+(913-0)+$
$(948-913)+(1022-948)+(1470-1022)$
$+(1509-1470)+(1750-1509)+(1774-1750)$

$= 13+44+86+913+35+74+448+39$
$+241+24$

$= 1917$

Average $= 1917/9$

$= 213$

11. Explain the following scheduling algo –

a) Shortest Remaining time next (SRTN)
(Preemptive SJF)

b) Time slice Scheduling (RRobin)

c) SJF (non preemptive)

2007

File protection in Unix

2008

1) Life cycle of thread
2) What is memory allocation method ?
3) Define single partition allocation and fixed equal multiple partition memory management system.
4) State bernsteim concurrency condition
5) Digital signature

2010

1) Critical Region
2) Interprocess communication
3) Virtual machine
4) Monitor
5) Raid

Critical Region

Critical Region in high level language synchronization construct requires that a variable b of type D. which is to be shared among many processes declared as v: shared D. The variable b can be accessed only inside a region of the following form. Region b when B Do S. This construct mean while statement S being executed no other process can access the variable b. The expression B is a boolean expression that control the access of critical region. When a process tries to enter the critical section region the boolean expression B is ~~executed~~ evaluated. If the ~~statement~~ expression is true the statement S is ^ execution. If the ~~statement~~ expression is false the process release the critical region.

Critical region construct guard against certain simple errors associated with the semaphore solution to the critical section

problem that may be made by programmer

Monitor - Another high level synchronization construct is monitor. A monitor is characterized by a set of programmer defined operator. The representation of monitor type consist of declaration of variables whose values define the state of the instance of a type as well as the bodies of the procedure or function. The representation of monitor cannot be used directly by the various processes. Therefore the procedure defined within a monitor can access only those variables locally within a monitor and ~~can access~~ its formal parameters. Similarly the local variables of a monitor can be accessed by only local procedures. The diagram of monitor is given below:

0 —— o ——

MD- 9874040410