

Operating Systems:

It is a system software that manages computer hardware, software resources, and provides common services for computer programs.

Detailed Contents:

Unit: 1 → Introduction to operating system:

Definition of operating system, Evolution of operating systems - simple batch systems, multi-programmed batch system, time sharing systems, functions of an operating system. single user & multiuser operating systems, Open-source & closed-source operating systems.

Unit 2: Process Overview

Def. of process, process states, process life cycle, Process control Block (PCB), Process scheduling - Scheduling queues, Schedulers (short term, medium term & long term), Dispatcher, Context switch.

Unit 3: CPU scheduling:

CPU scheduling, preemptive & non-preemptive scheduling, Scheduling criteria - CPU utilization, Throughput, Turnaround time, Waiting time, Response time. scheduling algorithms - first-come-first-serve, shortest-job-first, priority scheduling, Round-Robin.

Unit-4: Introduction of Deadlocks:

Normal mode of operation - Request - use - Release sequence, Def. of deadlock, Deadlock characterization, Necessary & sufficient conditions - Mutual exclusion, Hold & wait, No preemption and circular wait. Introduction of methods for handling deadlocks.

Unit-5: Memory Management Technique:

Fixed partitioning, dynamic partitioning, memory fragmentation, simple paging, simple segmentation, virtual memory with paging, virtual memory with segmentation, page fault, thrashing, Page replacement policies - FIFO, optimal, LRU

Unit-6: Storage Management

file concept - file attributes, file operations, file types, Access Methods - sequential access, direct access. Directory structure - directory overview, single-level directory, two-level directory, tree-structured directories. Disk storage Access ways - Host - Attached storage, Network - Attached storage, and storage Area Network, Disk scheduling - FCFS, SSTF, SCAN, C-SCAN

Unit - 1>:- Introduction to operating system. :-<

Operating System :- Def. An operating system is a program that manages the computer hardware it provides a basis for application programs and acts as an interface b/w the computer user and the computer hardware.

An operating system is a large and complex program. It must be created piece by piece. Each part of the operating system should be a well defined part of the system with well defined inputs, outputs and functions.

fig 1 :-

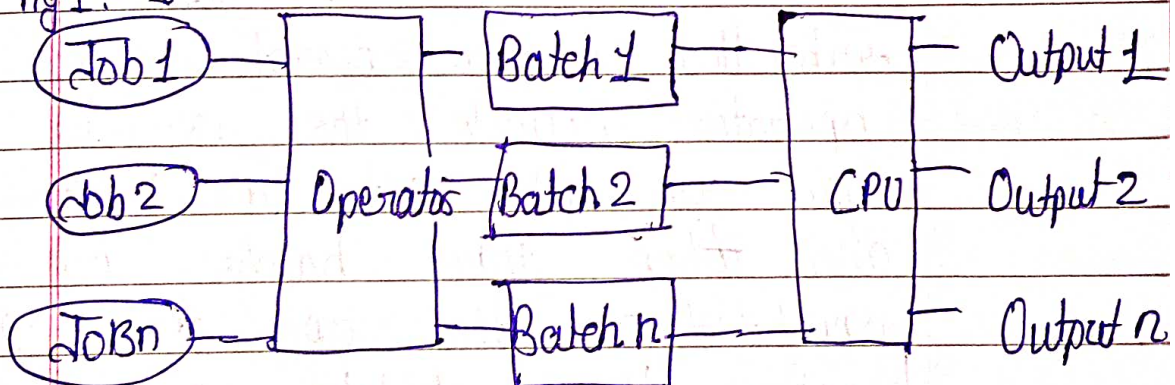
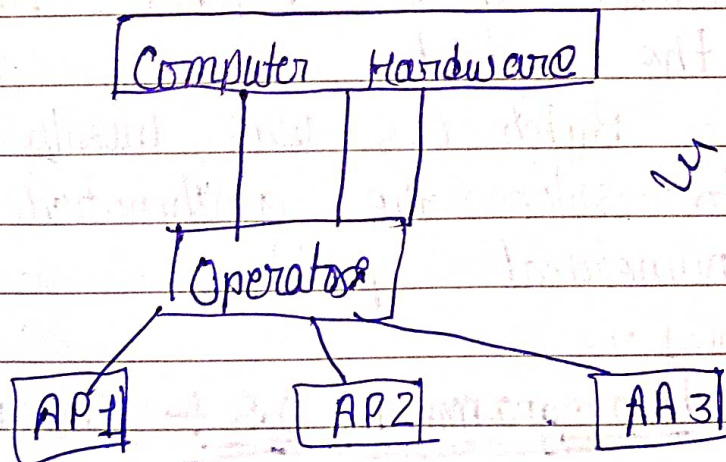


fig 2 :-



Evolution of operating system:-

Operating system has evolved over the time & now we have various types of operating systems with different features & functions.

Some of the operating systems are as follows:

1. Batch operating system.
2. Multiprogram Operating System.
3. Time sharing operating system.
4. Single user & Multi user O.S.
5. Open source & closed source O.S.

1. Batch Operating System :- In Batch O.S. the jobs are

submitted to the operator. The operator collects the similar types of jobs into a batch and then these batches are submitted to the CPU. The CPU solves these batches and gives the output.

Batch O.S. was mostly used to solve the arithmetic & numerical problems.

Fig :- 1.1

2. Multiprogrammed O.S :- In Multiprogrammed

D.S., the CPU works on more than one program at the same time. A computer system has many resources such as processors, input output devices, storage devices etc. It is not possible for the system to utilize all the resources effectively in a single program environment. Therefore when system is busy with input output operation of one program, the processor can operate on another program.

Therefore multiprogramming O.S. helps to increase the CPU utilization and use the source effectively.

3. Time Sharing O.S :- In time sharing O.S, a time quantum is decided and the programs are allocated the processor for this given time quantum. Time sharing O.S increases the CPU utilization by keeping the CPU busy all the time. Time sharing O.S helps to implement multiprogramming and multi-tasking environment.

4. Single user & Multi user O.S :- In a single user & Multi only one user can

perform all that system while the multi-user O.S. more than one user can share the CPU and other resources. Generally, CPU utilization is higher for Multi-user O.S.

Example of single user O.S :
DOS [Disk O.S]

Example of Multi user O.S :
Distributed O.S, Real time O.S etc.

5. Open source & closed source O.S :

In open source, the operating system is available free of cost along with its code. users can add, add features and redistribute the open source O.S.

In closed source O.S., the users have to pay fee to use the O.S. and the code is also not available. The users cannot make any changes in closed source O.S. Example of open sources are ubuntu, Linux, Kali, SUSE, Linux mint. Example of closed source are windows OS, Mac operating system.

Functions of O.S.

1. File Management
2. Storage Management
3. Protection & Security
4. Process Management
5. Memory Management
6. Device Management

23 Sep

1. File Management :- A file is a smallest unit of storage everything in a computer system needs to be stored in a file. A file can be of different types. Some examples of file type are doc file, pdf file, text file, audio file, video file, ppt file etc.

The type of file is identified using the extension of the file. Some popular file extensions are:

abc.doc - document file

abc.pdf - Portable document file

abc.ppt - Presentation file

abc.txt - text file.

Various operations can be performed on files. Some operations depend on the type of file. Some popular file operations are: Create, delete, update, truncate, concatenate, append etc.

2. Protection & Security : Operating system is also responsible for protection and security of a computer system. Various malicious programs and unauthorised user access are threats to the computer system. Computer system provides the protection & security to the computer system by using following softwares:

1. Firewall : A firewall acts as a wall b/w the computer system and internet. It scans the packets coming from the internet and rejects the packets if the content is not authentic.

2. Windows defender : It is a program used in windows operating system and it helps in proper functioning in windows O.S.

3. Anti-virus : An anti-virus also protects the computer system from many malicious programs.

3. Process Management : A process is defined as a program in execution. A number of operations are performed on a process and

these operations are performed by these operating system.

4. Memory Management :- Memory is used to store data & information on the computer system. Generally A memory is divided into two types?

1. Primary Memory
2. Secondary Memory

1. Primary Memory :- It has two types: ROM and RAM. RAM is used for executing the programs. It is the responsibility of OS to divide the RAM among various processes in a multi-programming environment. This is known as memory management.

5. Device Management :- (Resource management)

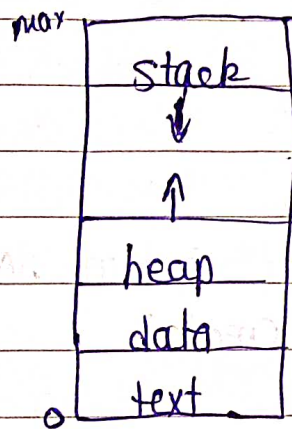
All the devices connected to a computer system are managed by O.S. The O.S. efficiently divides the devices among various processes and helps in proper working of multiprogram system. The O.S. receives the demand of devices from the process and allocate the devices to the process.

6. Storage Management :- The secondary memory of the computer system can store huge amount of data. It is the responsibility of the O.S to define the structure of the storage, access ways to the secondary storage etc.

UNIT-2Process Management

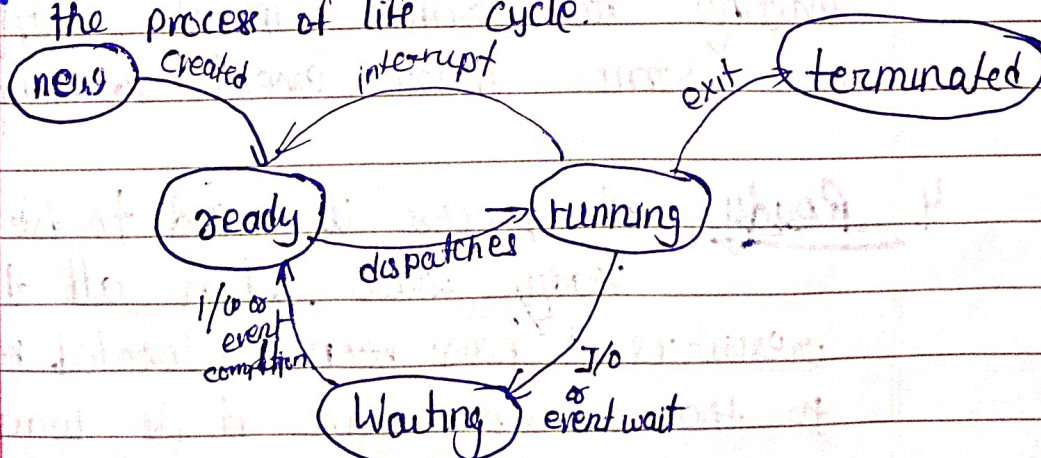
A process is a program in execution.
 a process includes program code, program counter, stack, various registers etc.

A program is a passive entity.
 [just an executable file stored on a disk]
 Whereas a process is a active entity

Process in memory

Program Counter stores the address of next instruction to be executive.

Process Life Cycle The following diagram represents the process of life cycle.



Process States: When a process executes it changes its states.

The state of a process is defined by the current activity of that process. Each process may be in one of the following states.

1. New
2. Running
3. Waiting
4. Ready
5. Terminated

1. New: A process is in new state when it is created.

2. Running: A process is in running state when it is allocated all the resources & instructions are being executed.

3. Waiting: A process is said to be in waiting state when it is waiting for some input, output or some other event to happen.

4. Ready: A process is said to be in ready state when all the resources have been allocated to the process and it is waiting

only for the processors CPU.

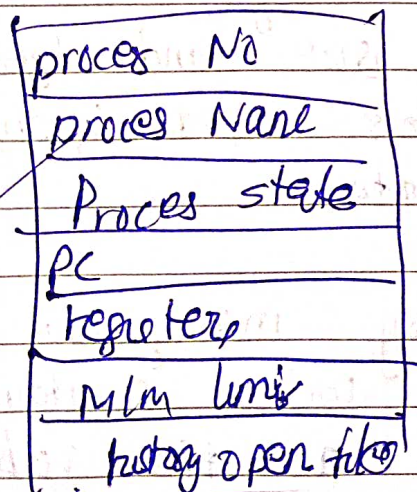
5. Terminated: A process is said to be in terminated state when it has finished its execution and it releases all the allocated resources.

Process life cycle

The following diagram represents the process life cycle.

Process Control Block (PCB)

Each process is represented in the operating system by a process control block. A PCB is also known as task control block as it contains many pieces of information associated with the process. The following diagram represents process control block.



1. Process No - Every process is given a unique no so that the two processes with the same name can be identified uniquely.
2. Process Name - It represents the name of the process.
3. Process state - This portion of PCB represents the current state of the process. A process may be in new, ^{ready} ready, running, waiting, terminated state.
4. Program Counter - It contains the address of next instruction to be executed.
5. CPU registers - A process makes use of various registers such as accumulators, stack register, index registers etc. The size and type of these registers depends upon computer architecture.
6. Memory management information - This portion contains the information such as the value of ^{base} ~~data~~ &

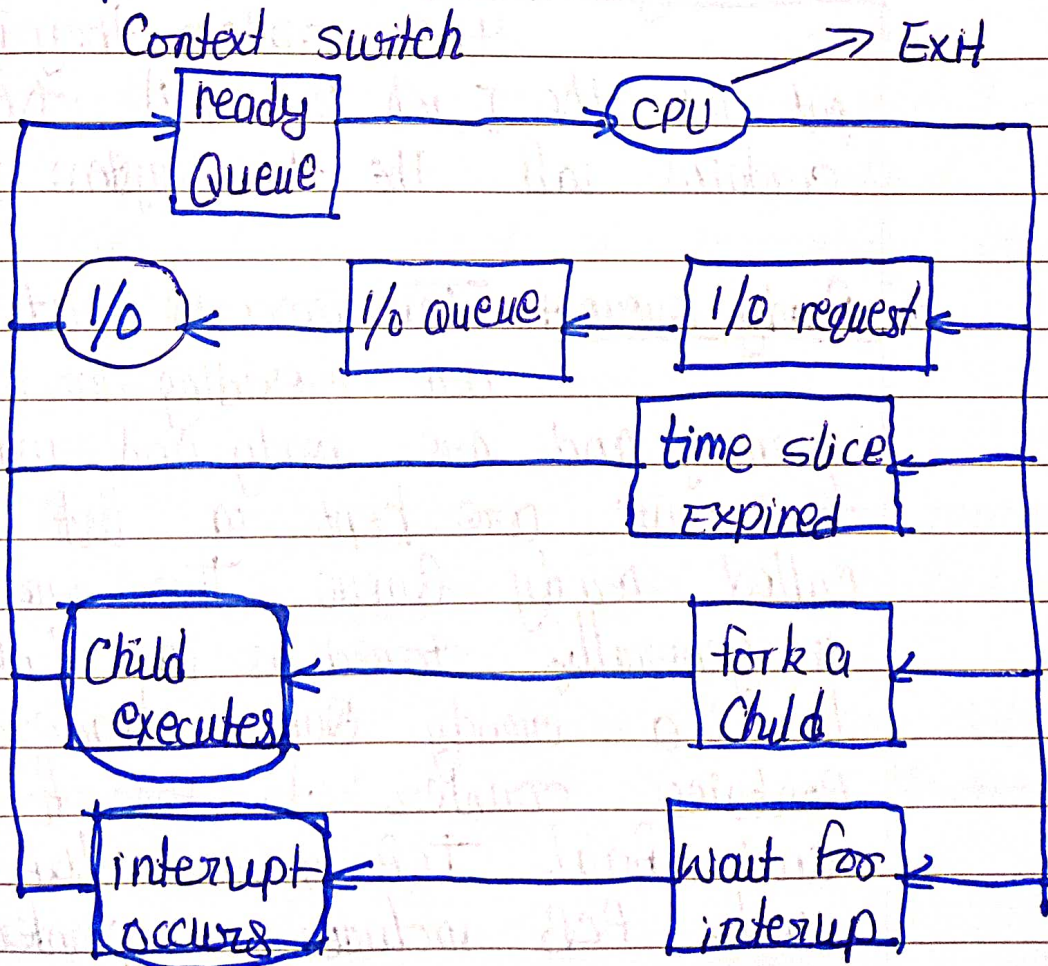
limit registers, page tables, segment table etc.

7. Input/output information: This information includes the list of input/output devices allocated to the process, list of open files etc.

Process scheduling:

Scheduling queues
Schedulers { short term
Medium term
Long term.

Dispatcher



Process Scheduling ÷ The objective of multi-programming is to have some process running at all times, to maximize CPU utilization.

The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running to meet these objectives, the process scheduler selects an available process for execution on the CPU.

* Scheduling Queues: As process enter the system they are put into the [Job Queue] A job queue contains all the the system.

* Ready Queue ÷ The processes that are residing in main memory and are ready and waiting to execute are kept in list called ready Queue. This Queue is generally stored as a linked list, a ready Queue header contains pointer to the first and final PCB in the list. Each PCB includes a pointer that

points to the PCB in the ready Queue. The system also includes a the Queue. When a process is allocated the CPU it executes for a while and it quits is interrupted or waits for some particular event such as input, Output etc. The process therefore may have to wait for the I/O or event. The list of processes waiting for a particular I/O device is called device Queue & each device has its own device Queue.

A New process initial put in the ready Queue it waits there until it is selected for execution. Once the process is allocated the CPU and is executing, one of the following event can occur,

1) The process can issue an I/O request and then be placed in I/O queue

2) The process could create new sub process and wait for the termination of child process.

3) The process can be removed from the CPU as a result of time slice expiration and put back in the ready queue.

* In fact to convert the process from waiting state to ready state and is put back in the ready queue. A process continues in the cycle until it terminates and after termination it is removed from the queue and has its PCB and resources be allocated.

11 Oct. Scheduler:- A process migrates among various scheduling queues and the CPU has to select one of the process for execution. This selection is carried out by scheduler. We have 3 types of scheduler:

1. Long term scheduler.
2. Short term scheduler.
3. Medium term scheduler.

1. Long term scheduler: It is also known as the job scheduler. All the processes which are created are present in job queue. The long term scheduler

selects the processes from the job queue and put them in the ~~ready~~ ready queue. The processes in the job queue are of two types:

- CPU bound,
- I/O bound.

⇒ CPU Bound: The process which spend most of its time on CPU for execution and very little time for I/O are known as CPU Bound processes.

⇒ I/O Bound: A process which spends most of its time performing I/O and a very little time on CPU is known as I/O Bound process.

It is the responsibility of long term scheduler to select a proper mix of CPU bound and I/O Bound process.

120cl Therefore the long term scheduler decides how many process will enter the radical. The number of process in the radical defines the degree of Multi-programming. Therefore the long term scheduler decides the degree of multiprogramming.

2. Short term scheduler! All the processes in the ready queue are in ready state and ready for the CPU. One of the process from the ready queue is selected and allocated the CPU. and when this process releases the CPU due to some event then the CPU is given to the next process. This process is done by a short term scheduler. The short term scheduler selects one of the process to be allocated to the CPU.

The short-term scheduler uses scheduling algorithms to decide which process will be allocated the CPU.

It is also known as CPU Scheduler.

3. Medium term Scheduler! Sometimes removing the processes from the ready queue may improve the degree of multiprogramming. Improving degree of multiprogramming leads to more CPU utilization. The decision of removing the process from the ready queue is taken by medium term scheduler.

Context switch: In multiprogramming the CPU is shared by number of processes. The interrupts cause the operating system to change the CPU from one process to another. When an interrupt occurs the system needs to save the current context of the process running currently on the CPU so that it can be restored on resuming its execution. This context is represented by PCB of that process which includes the value of CPU registers, the process state, memory management information etc.

Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch.

When the context switch happens, the old context is saved in the PCB of the old process and loads the saved context of the new process.

Context switch time is a pure overhead because the system is not doing any useful work while

switched.

The speed of context switch varies from machine to machine depending on the memory speed, no. of registers that must be copied etc.

DISPATCHER

Another component involved in the CPU scheduling function is the dispatcher. The dispatcher is the module that gives control of the CPU to the process selected by the short term scheduler. The function involves the following:

1. Switching context.
2. Switching to user mode.
3. Jumping to the proper location in the user program to restart the program.

The dispatcher should be as fast as possible. The time taken by the dispatcher to stop one process and start another process running is known as the dispatch latency.

Unit - 3

→ CPU Scheduling : (short term scheduling)

Preemptive and Non-preemptive Scheduling :

- FCFS
- SJF
- Priority sch.
- Round Robin [R.R.]

→ Preemptive scheduling : It is defined as the type of scheduling in which the CPU can be taken from the process during its execution. eg of preemptive scheduling are priority sch. and Round Robin [R.R.] scheduling.

→ Non-Preemptive scheduling : Non-preemptive scheduling means the resources and the CPU are not taken from the process during its execution. The process releases the resources and the CPU only after completing its execution. The examples of non-preemptive scheduling are.

1. FCFS.
2. SJF.

Scheduling criteria $\Rightarrow$ Different CPU scheduling algorithms have different properties. Selecting which algorithm to use, we must consider the properties of use, we must consider the properties of various algorithms. Following are the criteria for comparing various algorithms. Following are the criteria for comparing various CPU scheduling algorithms.

- 1) CPU utilization
- 2) Throughput
- 3) Turnaround time
- 4) Waiting time
- 5) Response time

i) **CPU utilization** $\div$ CPU utilization defines how much busy is the CPU. The value of CPU utilization can vary from 0 to 100, in real system CPU utilization is between 42 to 90%.

ii) **Throughput** $\div$ If the CPU is busy executing the processing, then work is being done. Throughput is used to measure work. Throughput defines as the no. of the process data are completing per unit time. For example, For long

process per are and for short process throughput may be 10 process per second.

Turnaround time ÷ The time interval from the time of submission of a process to the time of completion is define as the turnaround time.

Turn around time is some of time step waiting to get in to memory to in ready queue executing on the CPU and performing input output.

Waiting time ÷ It is define as the some of the time spend waiting in the ready queue.

Response time ÷ In many system time may not me the best criteria. In such system response time is used as scheduling criteria. response time is time taken by the system to give first response the user.

It is desirable to maximize CPU utilization and Throughput and minimize turn around time & waiting time and response time.

Scheduling Algorithm:

First come first serve [FCFS] + FCFS is the simplest CPU scheduling Algorithm. In this Algorithm, the process that request the CPU first is Algorithm the CPU first.

Considered the following the example

Process	cpu burst
P ₁	24
P ₂	3
P ₃	3

The following Gantt chart respect the execution these process using FCFS

P ₁	P ₂	P ₃
0	24	27 30

① Calculating avg. Turnaround time →

$$\text{avg turnaround time} = \frac{\text{turnaround } P_1 + \text{turnaround } P_2 + \text{turnaround } P_3}{\text{Total no. of process}}$$

$$\text{turnaround time of } P_1 = 24$$

$$\text{Turnaround time of } P_2 = 27$$

$$\text{Turnaround time of } P_3 = 30$$

There avg. turnaround time = $\frac{24+27+30}{3}$
 $= \frac{81}{3} = 27$

Calculating avg waiting time $\rightarrow$

~~$$\text{Avg WT} = \frac{\text{WT of } P_1 + \text{WT of } P_2 + \text{WT of } P_3}{3}$$~~

Waiting time of $P_1 = 0$

Waiting time of $P_2 = 24$

Waiting time of $P_3 = 27$

avg. waiting time $k = \frac{0 + 24 + 27}{3} = \frac{51}{3} = 17 \text{ hr}$

Convay effect + In FCFS Algorithm, when all the short process is wait for one big process to complete its execution then it is known as convay effect.

D) FCFS + scheduling Algorithms is now pointing. Once the CPU is being allocated to a process, that process keeps the CPU until it complete its execution.

(ii) **SJF [shortest Job first]**:- In this Algorithm, the CPU is

given to the process which has the smallest CPU burst time. If there is a conflict between two processes, FCFS Algorithm is used to break the tie.

H.T.T P₃

Consider the following example:-

Process

cpu burst

A

30

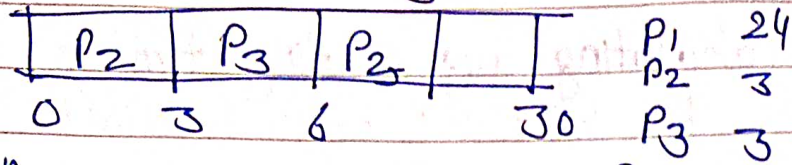
P2

2

P_3

6

$$\text{Avg TT} = \frac{P_1 + P_2 + P_3}{3} =$$



$$\text{Avg TT} = \text{Turn time } \frac{P_1 + P_2 + P_3}{3}$$

$$\frac{30 + 3 + 6}{3} = \frac{39}{3} = 13$$

$$\text{Avg Turn time} = \frac{\text{WT of } P_1 + P_2 + P_3}{3}$$

$$= \frac{6 + 0 + 3}{3} = \frac{9}{3} = 3$$

ii Example process CPU Burst time

 P_1

6

 P_2

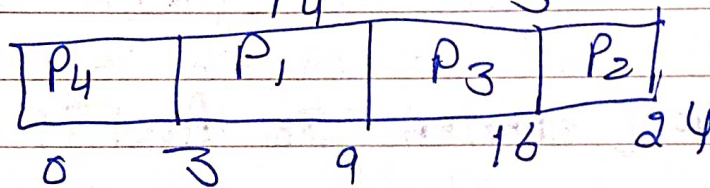
8

 P_3

7

 P_4

3



$$\begin{aligned} \text{Avg turn around time} &= \frac{\text{I.T of } P_1 + P_2 + P_3 + P_4}{4} \\ &= \frac{9 + 24 + 16 + 3}{4} = \frac{52}{4} = 13 \end{aligned}$$

$$\text{Avg. waiting (WT) time} = \frac{\text{W.T. of } P_1 + P_2 + P_3 + P_4}{4}$$

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

2

process	cpu Burst time
P ₁	6
P ₂	8
P ₃	7
P ₄	3

SJF [shortest job first]

P ₄	P ₁	P ₃	P ₂
0	3	9	16

Avg Turn around time = $\frac{\text{TT of } P_1 + \text{TT of } P_2 + \text{TT of } P_3 + \text{TT of } P_4}{4}$

$$= \frac{9 + 24 + 16 + 3}{4} = \frac{52}{4} = 13$$

Avg. waiting time = $\frac{\text{WT of } P_1 + \text{WT of } P_2 + \text{WT of } P_3 + \text{WT of } P_4}{4}$

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

(iii) Priority scheduling +

Process	cpu time	Priority
P ₁	10	3
P ₂	1	1
P ₃	2	4
P ₄	1	5
P ₅	5	2

⇒ In priority scheduling Algorithm
a priority is associated with
each process and the CPU is
allocated to the process with
Algorithm priority. Some system use
low numbers to represent low
priority, others use low numbers
for high priority.
Consider the following example:
priority scheduling!

P_2	P_5	P_1	P_3	P_4	
0	1	6	16	18	19

Round Robin scheduling:

Example: time quantum = 4

P ₁	24-26-16-12-8	P ₁	P ₂	P ₃	P ₁	P ₁	P ₁	P ₁
P ₂	3	0	4	7	10	14	18	22
P ₃	3							26
								30

$$\text{Avg time} = \frac{\text{TT of } P_1 + P_2 + P_3}{3} = \frac{30 + 7 + 10}{3} = \frac{47}{3} = 15.66$$

$$\text{Avg time T} = \frac{\text{W T of } P_1 + P_2 + P_3}{3} =$$

$$\frac{6 + 4 + 7}{3} = \frac{17}{3} = 5.667$$

Round Robin scheduling algorithm is designed specially for time sharing system. It is similar to first come first scheduling system but preemption is added to switch b/w processes. A small unit of time called a time quantum or time slice is defined. A time quantum generally ranges from 10 to 100 ms.

The performance of round robin algorithm depends heavily on the size of the time quantum. If it is very large then the round robin algorithm is

same as first come first serve.
If the time quantum too small
then the system spends its
significant amount of ^{time} in
context switching.

Starvation :

One of the major problem
with priority scheduling is
starvation. It is also known
as indefinite blocking.

In priority scheduling in
low priority process may have
to wait indefinitely because
of high priority processes.
The solution to this problem
is aging.

Aging is a technique of
gradually increasing the priority
of a process that wait in a
system for a long time.

Unit - 4

Normal mode of operation :

- Request
- Use
- Release

A system consist of ~~infinite~~ a finite number of resources to be distributed among a number of competing processes. The resources are partitioned into several types, each type consisting of some number of identical instances.

Memory space, CPU cycles, files, input/output devices are examples of resource type. If a system has two printers then the resource type printer has two instances.

A process must request a resource before using it and must release the resource after using it. A process may request as many resources as it requires to carry out its task. The number of resources requested may not exceed the total no. of resources available in the system.

In a normal mode of operation

a process may utilize. The resource in the following sequence.

1. Request: A process first make a request to the operating system for required resources.

The process can use the resources only after it is allocated to that process. If the request is not granted then the process may have to wait until the resource is allocated to it.

2. Use: The process can operate on the resource.

3. Release: After using the resource, the process releases all the resources.

11 Nov. Deadlock Characterization.

~~No~~

Deadlock!

X Necessary & sufficient conditions for deadlock

- Mutual exclusion
- Hold & wait
- No preemption
- Circular wait.

Definition of Deadlock: A set of processes is in deadlock state when every process in the set is waiting for an event, that can be caused only by another process in the set. The event which generally causes deadlock is: resource acquisition & release.

In a deadlock, a process never finishes execution and system resources are tied up, preventing other processes from execution.

Deadlock characterization :-

- * DC
1. → Necessary & sufficient condition for deadlock
 2. → Resource allocation graph.

1. Necessary & sufficient condition for deadlock:-
A deadlock situation may occur in a system if the following four conditions hold simultaneously in a system.

- (i) Mutual exclusion
- (ii) Hold & wait
- (iii) No preemption
- (iv) Circular wait

(i) **Mutual exclusion** :- At least one resource in a system must be held in non-shareable mode i.e. only one process at a time can use the resource. If another process request that resource, the requesting process must be delayed until the resource has been released.

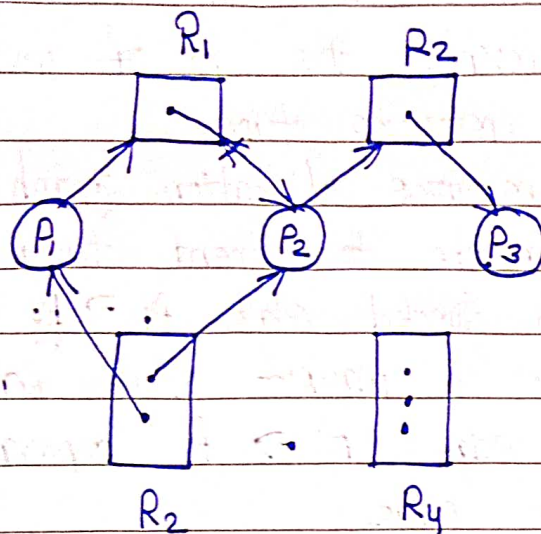
(ii) **Hold & Wait** :- A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

(iii) **No preemption** :- Resources can't be preemptive i.e. a resource can be released ~~voluntarily~~ voluntarily by the process holding it after completing its execution.

(iv) **Circular wait** :- A set of $\{P_0, P_1, P_2, \dots, P_n\}$ A waiting processes must exist such that P_0 is waiting for a resource held by P_1 , P_1 is waiting for a resource held

by P_2 and so on and P_n is waiting for a resource held by P_0 .
All the four conditions must hold for a deadlock to occur.

Resource Allocation Graph:

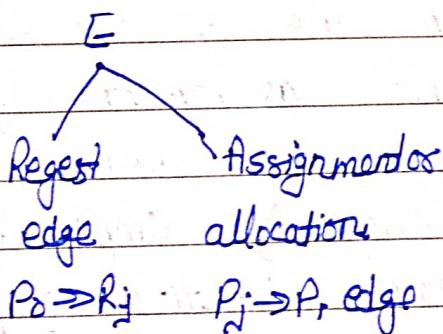


$$P = \{P_1, P_2, P_3\}$$

$$R = \{R_1, R_2, R_3, R_4\}$$

$$\{P_1 \rightarrow R_1, P_2 \rightarrow P_3, R_1 \rightarrow P_2, P_3 \rightarrow P_3,$$

$$R_2 \rightarrow P_1, R_2 \rightarrow P_2\}$$



cycle + only one un-

R_1

$R_2 - 1$

$P_3 - 2$

$P_4 - 3$

Deadlocks can be described more precisely using directed graph called resource allocation graph. The graph consist of set

of vertices and a set of edges. The resource allocation graph has two different types of vertices. The circular vertices represent the processes & the rectangular vertices represent the resource types of the system. The number of dots in the resource type represents the no. of instances of that resource type.

In resource allocation graph the edges also represent to different types of edges. A directed edge $P_i \rightarrow R_j$ is called as a request edge and a directed edge $R_j \rightarrow P_i$ represents an assignment edge.

Note that a request edge points to only the rectangle R_j , whereas an assignment edge must also designate one of the dots in the rectangle.

When a request for a resource is fulfilled then the request edge is transformed into assignment edge.

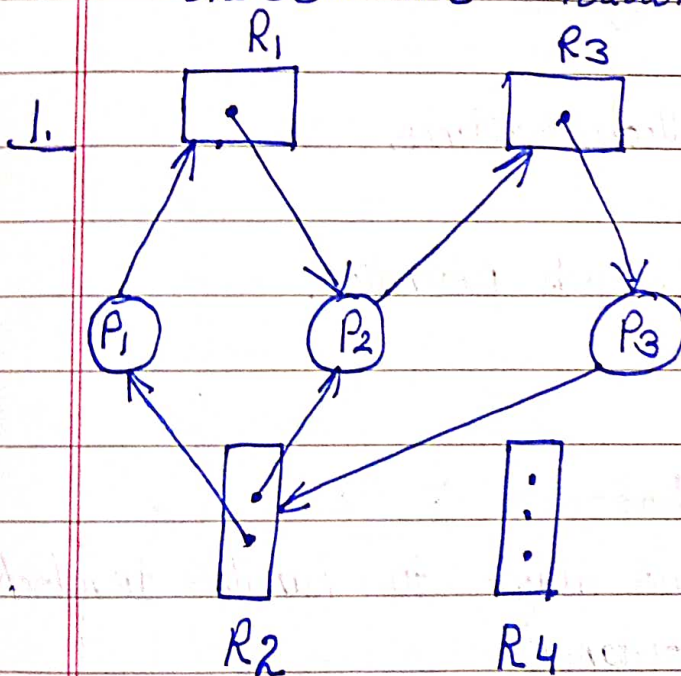
When the resource is released then the assignment edge is simply deleted from the resource allocation graph.

The resource allocation graph can be used to define the following:

1. If the graph contains no cycles then

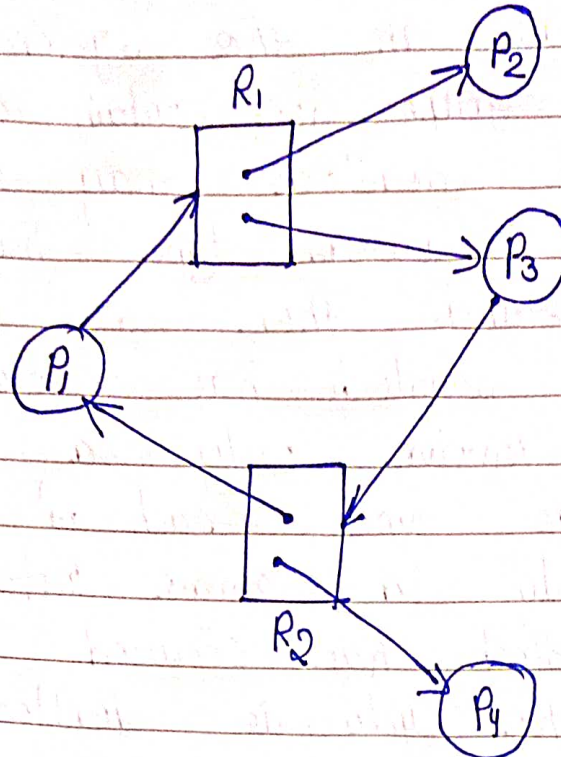
- no process in the system is deadlocked
2. If the graph does contain a cycle then a deadlock may exist. If at each resource type has exactly one instance then a cycle implies that a deadlock has occurred. If the cycle involves only a set of resource types, each of which has only a single instance then a deadlock has occurred. Each process in the cycle is deadlocked.
3. If each resource type has several instances, then a cycle does not necessarily imply that a deadlock has occurred.

Consider the following examples:



RAG with a cycle and a
deadlock.

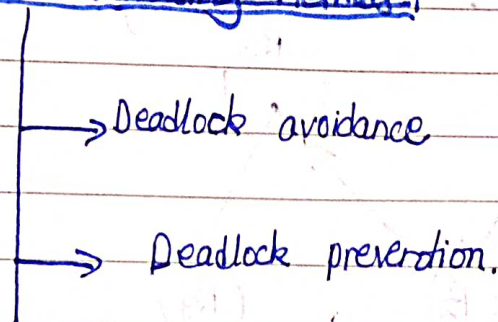
2.



Resource allocation graph with a cycle but no deadlock.

15 Nov.

Deadlock Handling Methods:-



Deadlock Handling

There are two ways to handle deadlocks:

1. deadlock avoidance
2. » prevention.

1. Deadlock avoidance - It requires that the O.S be given in advance.

Additional information concerning which resources a process will request and use during its lifetime with all the information the system can decide the correct way to allocate resources so that the deadlock does not occur.

2. Deadlock prevention - It provides a set of methods for ensuring that at least one of the necessary conditions can not hold.

The condition can be preventive in following ways:

1. Mutual condition: This condition can be preventive by making the resource sharable but it is not always possible to make resources sharable because some resources are intrinsically non-sharable. In this situation the no. of instances of that resource can be increased but it is not very ~~caused~~ cost efficient.

2. Hold & Wait: This condition can be prevented by guaranteeing by that whenever a process request a resource

it can't hold any other resource

In other words a process must request for all resources in the beginning and these resources should be allocated before the beginning of execution.

16 Nov.

3. No preemption:- The third necessary condition for deadlock is that there be no preemption of resource that have already been allocated. 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 can not be immediately allocated then all the resources currently been held ~~it~~ are preempted.

4. Circular wait:- One way to ensure that this condition never holds is to impose a total ordering of all resource types and to require that each process request resources in an increasing order of number.

e.g. consider a set of resource type are
 $R = \{R_1, R_2, R_3, \dots, R_n\}$

We assign to each resource type a unique number which allows us to compare two resources and to determine whether one precedes another in our ordering. Now a process can request resources only in an increasing order of number.

A process can request any number of instances of a resource type ~~are~~ R_i after that process can request instances of resource type R_j if and only for $F(R_j) > F(R_i)$ where f is a 1-1 function such that
 $F: R \rightarrow N$

↓ ↓ ↓ Natural number

function Resource type

If several instances of the same resource type are needed then the process must send a single request. Using this protocol we can prevent circular wait.

18 NOV.

Unit - 5

Memory Management ÷ It can be partitioned into various parts by using one of the following methods:

1. fixed partitioning

2 Dynamic Partitioning

① fixed partitioning: In fixed partitioning the memory is divided into fixed sized partitions. the size of the memory block is fixed and all the blocks in the memory are of same size. In this method these fixed size blocks are allocated to the process and if the required size is greater than the block size then more blocks are allocated in continuous order.

It is simple to implement but it suffers from fragmentation: fragmentation is generally of two types:

1. Internal fragmentation
2. External fragmentation

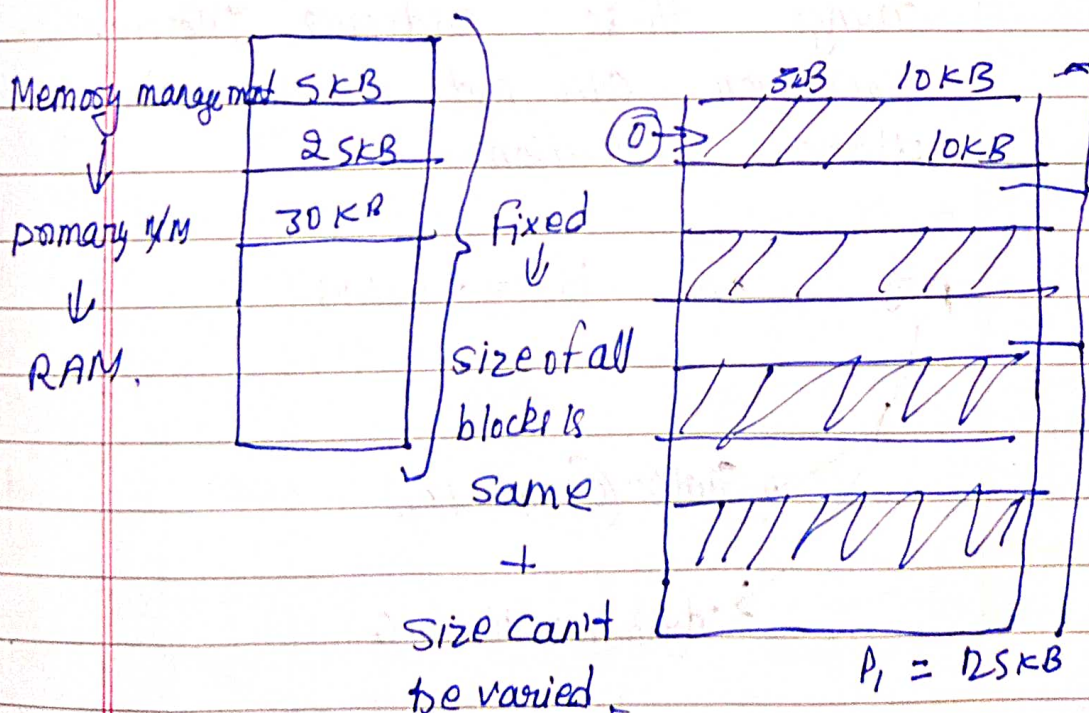
1. Internal fragmentation: It occurs when the size of the memory block is greater than a size of the memory required by the process.

eg:- If 10 KB is the size of fixed memory blocks and a process requires 5 KB then the

remaining 5 KB is wasted. This is known as Internal fragmentation.

2 External fragmentation $\rightarrow$ It occurs when enough memory is available but it is not continuous. Therefore it can not be allocated. In this situation enough memory is available but it can't be allocated.

eg:- A process requires 25 KB of memory and three blocks of 10 KB are available but these cannot be allocated because the blocks are not in continuous order. This is known as External fragmentation.



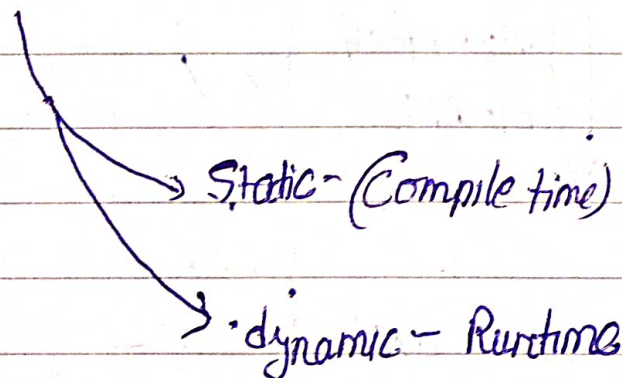
30 Nov.

Compaction +

It acts as a solution for external segment fragmentation. It is a process in which all the allocated blocks of the memory are put together or move to one side of the memory and all the free blocks are moved to the other side.

It seems a good solution but it only works for dynamic partitions allocation because the memory addresses are not linked to the code. In static allocation the memory addresses are linked with a code at compile time and it is not possible to change these addresses. Therefore compaction can not occur in static allocation.

prog m/m linkers & loaders



Paging ÷

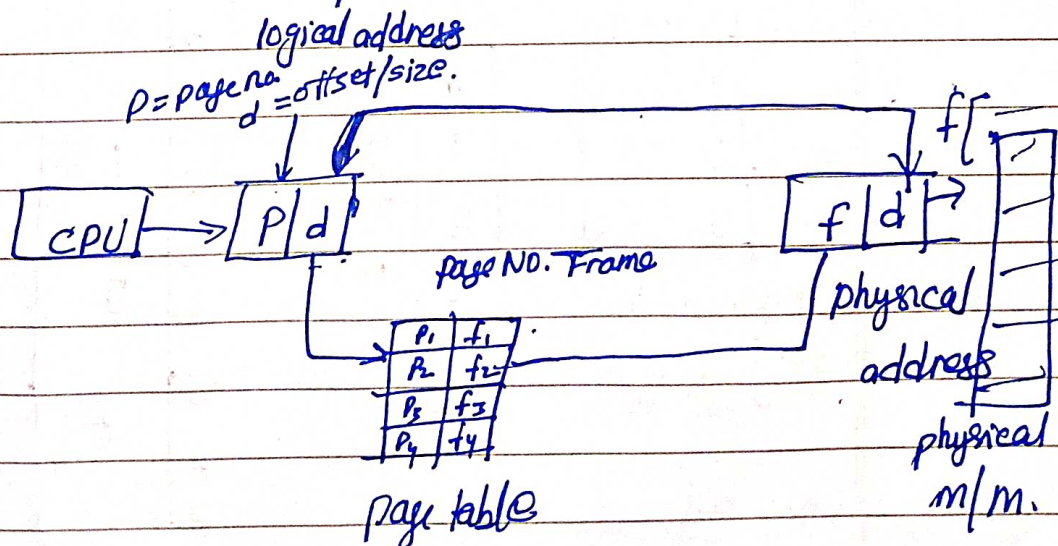
→ Physical address space

m/m unit.

→ Logical address space

Paging is a memory management scheme that permits the physical address space to be non-contiguous. The basic method for implementing paging involves breaking physical memory into fixed sized blocks known as frames and breaking the logical memory into blocks of same size known as pages. When a process is to be executed its pages are loaded into any available memory space frame from backing store. The backing store is divided into fixed sized blocks that are of the same size as the memory frames.

The hardware support for paging can be represented as follows:



The paging is model of physical or memory or logical memory. implementation of physical or logical memory.

The CPU generates the logical address which is divided into two parts: page no. or page offset.

page no. is used for looking into page table for required frame. The page table contains the base address of every frame. page off

Page offset i.e. is added to this base address to generate physical address.

Advantages of paging

26 Nov

Representation of paging in m/m

page 0
1
2
3

program in
logical m/m

0	6
1	2
2	0
3	4

page table

0
1
2
3
4
5
6
7

physical m/m

The program divided in pages and the physical memory divided into frames can be mapped easily using a page table. The page table contains two columns. One column specifies the pages of the program and the other column specifies on which frames in physical memory those pages are saved.

Paging with virtual memory: Virtual memory is a technique for memory management by which huge amount of programs or large programs can be executed using small size of physical memory. When we implement virtual memory with paging the program is divided into pages and the physical memory is divided into frames. But instead of mapping all the pages onto the frames, only a small no. of frames are allocated to the program. All the pages share those frames by swapping out a page by and swapping in the required page.

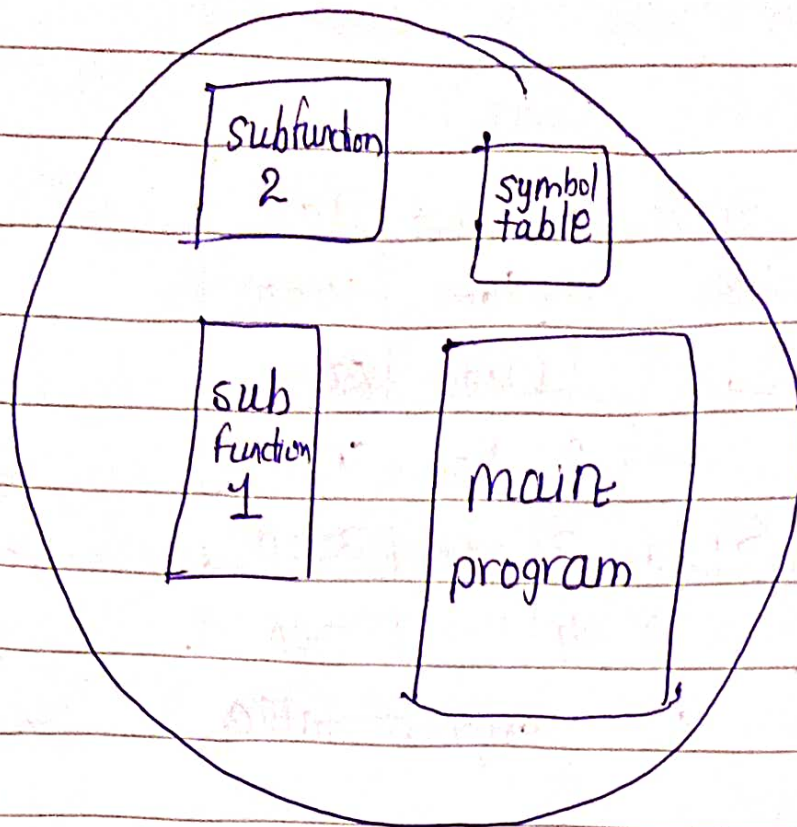
The no. of frames to be allocated

to the program are calculated using various algorithms such as working set model, etc.

Segmentation:-

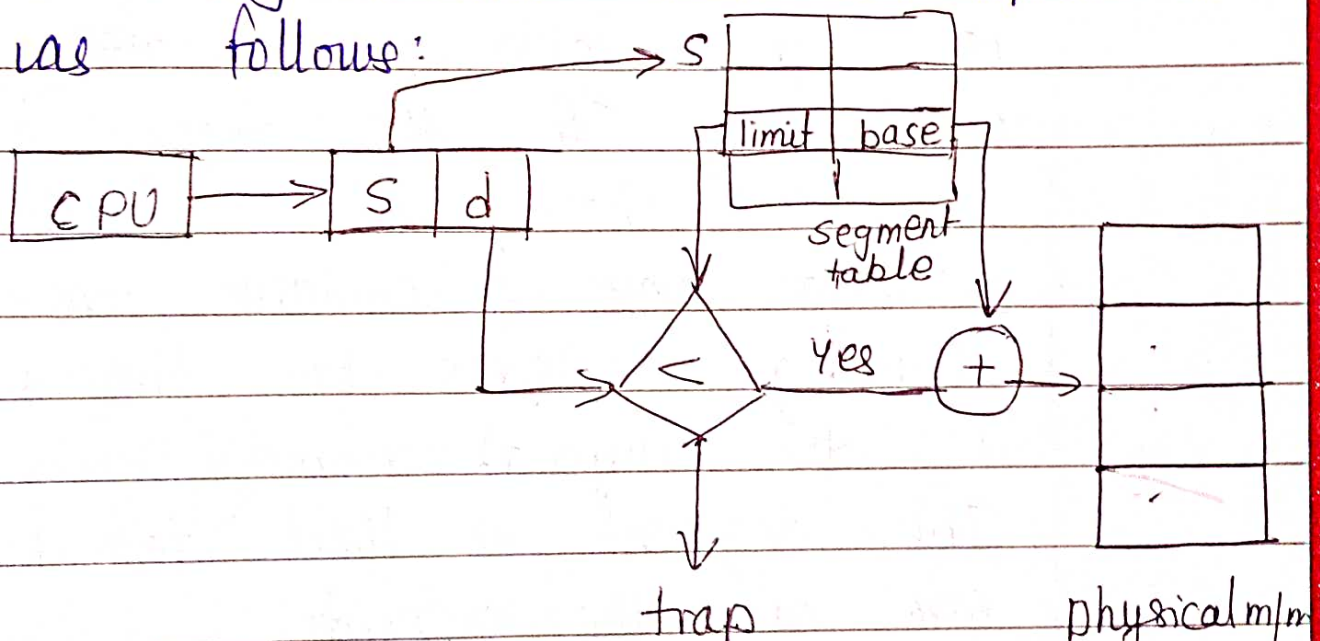
Paging was used to give a solution for external fragmentation but paging has a drawback. It is not practically possible to divide a program in fixed sized blocks. e.g. If dividing a program into a fixed sized paging may separate a function into two different pages and this function will not work properly or work at all in both the pages. To provide a solution to this problem segmentation was used.

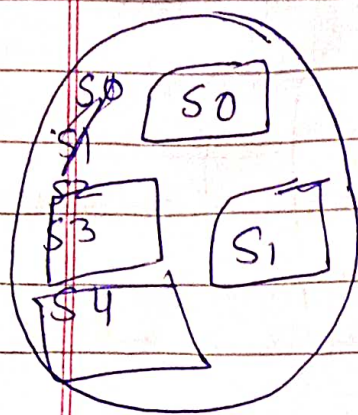
Segmentation is a memory management technique that supports the user view of memory. User view of memory means dividing the program into segments and the segments are logical units of the program.



Program

Each segment has a name and a length. The hardware implementation of segmentation can be represented as follows:





id	limit	Base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1100	4700

segment table

240	S0
320	S3
4300	
4700	S2
5700	4
6300	
6700	S1
	physical m/m

The address generated by CPU contains two tuples. $\langle S, d \rangle$ S is segment no and d defines the offset/size.

A segment table is generated and each entry in the segment table has a base and a limit. The segment base contains the starting address of the segment in the physical memory.

The segment limit defines the size of the segment.

20ec Page fault - When a process can not find the page in the memory then a page fault is set to occur. When a page is found it is known as

page hit.

Thrashing - When a page fault occurs then some page is replaced with the required page. A page in memory is swapped out and the required page is swapped in. When the process spends more time swapping pages in & out i.e. its paging activity than execution, thrashing is said to be occurring if a process

Thrashing occurs when a process spends more time paging than execution.

Belady's anomaly - A page replacement algorithm is said to have belady's anomaly if the page fault rate increases by increasing the number of allocated frames.

3 Dec. Page Replacement Algorithms:

FIFO

optimal page replacement

LRU [least recently used]

① FIFO [first in First Out]

In this page replacement algorithm the page which came first is replaced first consider the following example:-

Example 1:-

string {
reference {

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	4	0	7	7	7	7	7	7	7
	0	0	0	0	3	3	3	3	2	2	2	1	1	0	0	0	0	0	0
		1	1	1	1	0	0	0	3	3	3	2	2	2	2	2	2	2	2

page fault - ||||| = 15

Example 2:- suppose the allocated frames are four

7	0	1	2	0	3	0	4	2	3	0
7	7	7	7	7	3	3	3	3	3	3
	0	0	0	0	0	0	4	4	4	4
		1	1	1	1	1	1	1	1	0
			2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1

3	2	1	2	0	1	7	0	1
3	3	2	2	2	3			
4	4	7	4	4	4			
0	0	0	0	0	0			
2	2	1	1	1	1			
1	1	1	1	1	1			

page fault - ||||| = 10

Example :-

string refence: 1 2 3 4 1 2 5 1 2 3 4 5

1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4
1	1	1	1	1	1	1	1	1	1	1	1

Page fault = 11111111 = 9

1 2 3 4 1 2 5 1 2 3 4 5

1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
1	1	1	1			1	1	1	1	1	1

page fault = 10

First in First out [FIFO] algo page replacement algorithm suffers from belady's anomaly. i.e. by increasing no. of allocated frames, it is not necessary that page fault rate reduces.

6 Dec

② Optimal page replacement algorithm :-

string 7 0 1 2 0 3 0 4 2 3

ref

7	7	7	2	2	2	2	2	2	2
	0	0	0	0	0	0	4	4	4
		1	1	1	3	3	3	3	3
1	1	1							

0	3	2	1	2	0	1	7	0	1
2	2	2	2	2	2	2	7	7	7
0	0	0	0	0	0	0	0	0	0
3	3	3	1	1	1	1	1	1	1

page fault $\Rightarrow$ 11111111 = 9

Example 2 $\div$

7	0	1	2	0	3	0	4	2	3
7	7	7	7	7	3	3	3	3	3
	0	0	0	0	0	0	0	0	0
		1	1	1	1	1	4	4	4
			2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1

0	3	2	1	2	0	1	7	0	1
3	3	3	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0
4	4	4	4	4	4	4	7	7	7
2	2	2	2	2	2	2	2	2	2
			1				1		

page fault $\div$ 11111111 = 8

An optimal page replacement, we replace the page that will not be used for longest period of time. This algorithm has the lowest page fault rate among all the algorithms, also by increasing no. of allocated frames, the page fault

rate always decreases. therefore
optimal page replacement algorithm
never suffers from belady's
anomaly.

But optimal page replacement
algorithm is difficult to implement
because it requires ~~few~~ future
knowledge of string reference.

3. Least recently used (LRU) page replacement algorithm.

7	0	1	2	0	3	0	4	2	3
7	7	7	2	2	2	2	4	4	4
	0	0	0	0	0	0	0	0	3
		1	1	1	3	3	3	2	2

0	3	2	1	2	0	1	7	0	1
0	0	0	1	1	1	1	1	1	1
3	3	3	3	3	0	0	0	0	0
2	2	2	2	2	2	2	7	7	7

$$\text{page fault} \div 1111111111 = 12$$

an LRU algorithm, we replace the
page that has not been used
for the longest period of time
This algorithm does not suffer from

below's anomaly. This algorithm requires hardware assistance of counters and stack for its implementation.

classmate

Date _____
Page _____