Mumbai Education Trust's

INSTITUTE OF ENGINEERING, BKC, NASHIK.

T.E (Computer Engineering)

SYSTEM PROGRAMMING & OPERATING SYSTEM (2019 Pattern)

END-SEM DEC-2022

Time : 2 1/2 Hour *[Max. Marks : 70]*

Instructions to the candidates:

1) *Solve Que. 1 or 2, Que.3 or 4, Que.5 or 6, Que. 7 or 8 .*
2) *Neat diagrams must be drawn wherever necessary.*
3) *Assume suitable data if necessary.*
4) *Figures to the right indicate full marks.* **Date : 17/01/2023**

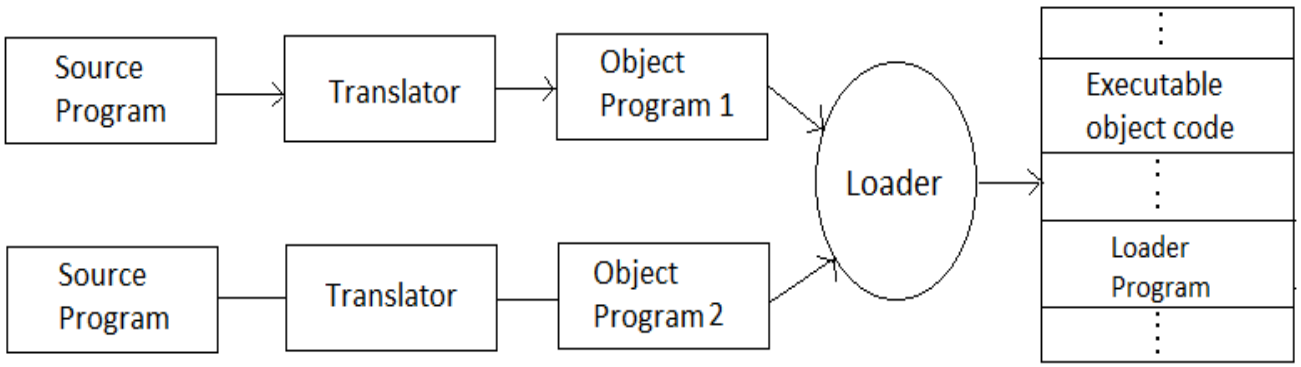
SPOS ENDSEM MODEL ANSWER DEC-2022

Q 1. a) Explain "General loading scheme(using suitable diagram)" with advantages and disadvantages. 9-Marks

Ans :- **General Loading Scheme :**

In this loader scheme, the source program is converted to object program by some translator (assembler). The loader accepts these object modules and puts the machine instruction and data in an executable form at their assigned memory. The loader occupies the same portion of main memory.

- In "Compile-and-Go", In which assembler is placed in main memory that results in wastage of memory
- To overcome that we requires the addition of the new program of the system, a loader.
- This **loader is assumed to be smaller than assembler** so that more memory is available to user.
- The loader accepts the assembled machine instructions, data and other information present in the object format and places machine instructions and data in core in an executable computer form.
- Reassembly is not necessary to run the program at later stage



Advantages:

- Smaller than assembler.
- No reassembly is needed.
- Possible to write subroutines in different languages.
- There is no wastage of memory, because assembler is not placed in the memory, so more memory is available to the user.
- This scheme avoids the disadvantages of preceding "compile-and-go" scheme.

Disadvantages:

- Some portion of the memory is occupied by the loader

b) Give complete design of Direct Linking Loader with suitable example. 9-Marks

Ans :

Direct Linking Loader is a general re-locatable loader

• Allowing the programmer multiple procedure segments and multiple data segments and giving programmer complete freedom in referencing data or instruction contained in other segments.

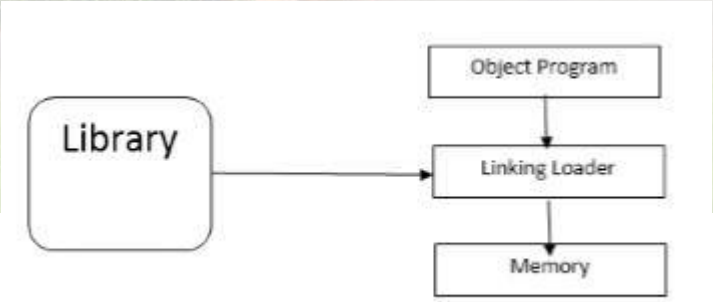
Dynamic linking defers much of the linking process until a program starts running. It provides a variety of benefits that are hard to get otherwise.

• Dynamically linked shared libraries are easier to create than static linked shared libraries.

• Dynamically linked shared libraries are easier to update than static linked shared libraries.

• The semantics of dynamically linked shared libraries can be much closer to those of unshared libraries.

• Dynamic linking permits a program to load and unload routines at runtime, a facility that can otherwise be very difficult to provide.



Length of segment

A list of all symbols in the segment that may be referenced by other segments.

List of all symbols not defined in the segment but referenced in the segment.

Information where the address constant are loaded in the segment.

Format of Databases:

The assembler provides following types of record in the object file as follows

External Symbol Dictionary (ESD)

• ESD record combine information about all the symbol that are defined in this program.

• But may be referenced in the program but defined elsewhere.

Text Cards (TXT)

• Text card record control the actual object code translated version of the source program.

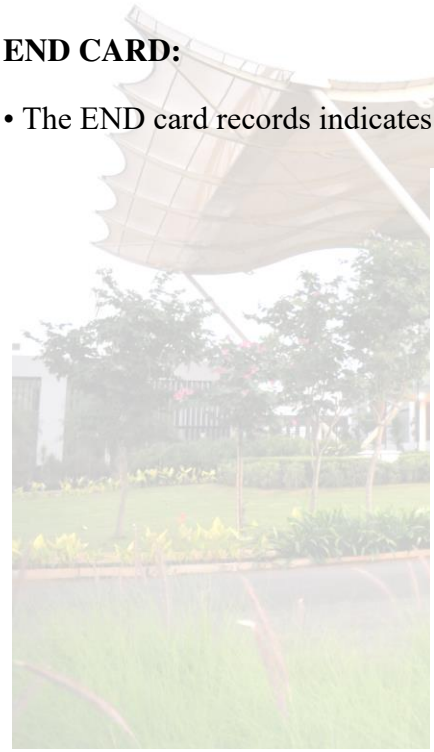
Relocation and Linkage Directory (RLD):

• The RLD records contains information about location in the program

• Whose contents depend on the address at which the program is placed.

END CARD:

• The END card records indicates the end of the object File and specifies the start address for Execution.



Example;

```
RA
0     PG1   START
4     ENTRY     A, B
8     EXTRN     PG2, C
        |
20    A     DC
24    B     DC...
        |
40    A + 20
44    B – 25
48    C – 5
|
52    END
```


ESD

IT consist of three types of definition
• SD- Segment Definition
• LD – Local Definition
• ER – External Reference

Symbol	Type	Id	RA	Length
PG1	SD	1	0	52
A	LD	1	20	
B	LD	1	24	
PG2	ER	2	-	
C	ER	3	-	

Relocating and Linking directory (RLD)
It includes different operations performed on it.
Flag is important in this table

Esd id	Symbol	Flag	R.A
1	A	+	40
1	B	-	44
3	C	-	48

END
• Specifies the END of the program. • And total address is generated by PLA + S Length.

OR

Q 2. a) Give Complete Design of Absolute Loader with suitable example ? 9-Marks

➔ **Data structure used :**

The absolute loader transfers the text of the program into memory at the address provided by the assembler after reading the object program line by line. There are two types of information that the object program must communicate from the assembler to the loader.

It must convey the machine instructions that the assembler has created along with the memory address.

It must convey the start of the execution. At this point, the software will begin to run after it has loaded.

The object program is the sequence of the object records. Each object record specifies some specific aspect of the program in the object module. There are two types of records:

Text record containing a binary image of the assembly program.

Transfer the record that contains the execution's starting or entry point.

The formats of text and transfer records are shown below:

Record Type	Number of bytes of information	Memory address	Binary image of data or instruction

Text record

Record Type	Number of bytes of information = 0	Address of the entry point

Transfer record

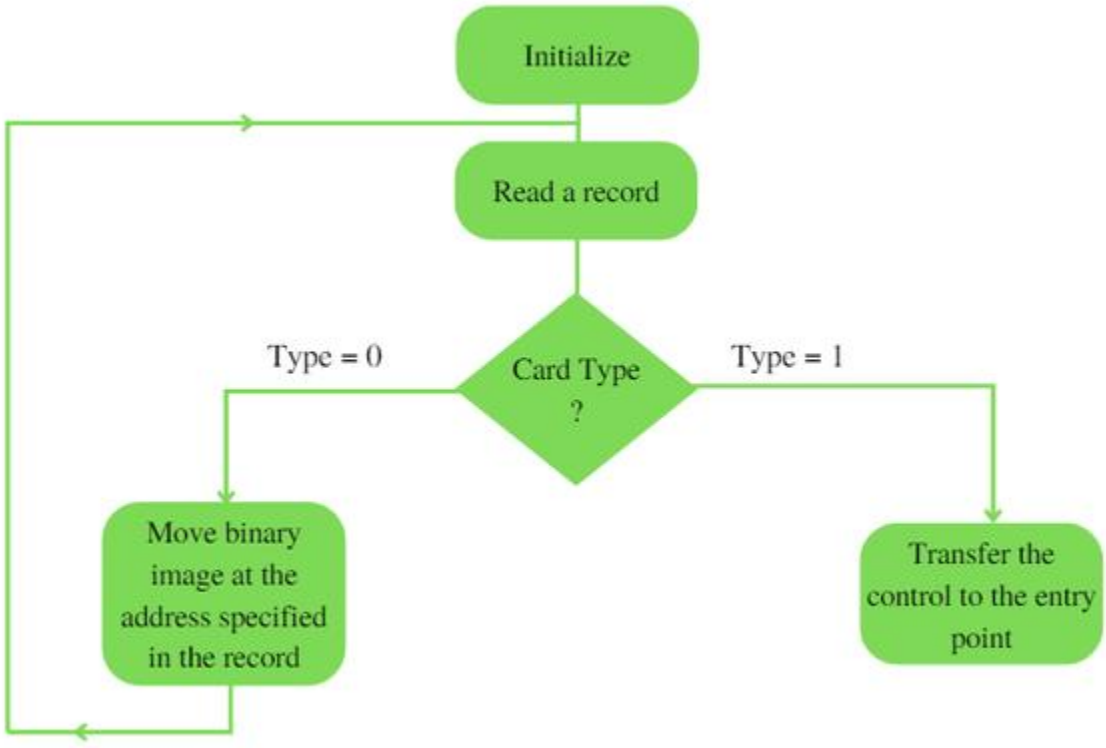
Record type = 0 for Text record
Record type = 1 for Transfer record

Formats of object records

Algorithm:

The algorithm for the absolute loader is quite simple. The object file is read record by record by the loader, and the binary image is moved to the locations specified in the record. The final record is a transfer record. When the control reaches the transfer record, it is transferred to the entry point for execution.

Flowchart :



Advantages:

1. It is simple to implement.
2. This scheme allows multiple programs or the source programs written in different languages. If there are multiple programs written in different languages then the respective language assembler will convert it to the language and common object file can be prepared with all the ad resolution.
3. The task of loader becomes simpler as it simply obeys the instruction regarding where to place the object code to the main memory.
4. The process of execution is efficient.

Disadvantages:

1. In this scheme, it's the programmer's duty to adjust all the inter-segment addresses and manually do the linking activity. For that, it is necessary for a programmer to know the memory management.
2. If at all any modification is done to some segment the starting address of immediate next segments may get changed the programmer has to take care of this issue and he/she needs to update the corresponding starting address on any modification in the source.

b) What is need of DLL? Difference between Dynamic and Static Linking. 9-Marks

Ans :-

- A dynamic link library (DLL) is a collection of small programs that larger programs can load when needed to complete specific tasks. The small program, called a DLL file, contains instructions that help the larger program handle what may not be a core function of the original program.
- An example of those tasks might be communicating with a specific device, such as a printer or scanner to process a document. DLL files that support specific device operations are known as device drivers.
- DLL contains bits of code and data, like classes and variables, or other resources such as images that the larger program can use.
- In addition to being a generic term for dynamic link libraries, Dynamic Link Library is also the name of Microsoft's version of the shared library concept for Windows. A shared library can exist in any operating system (OS).

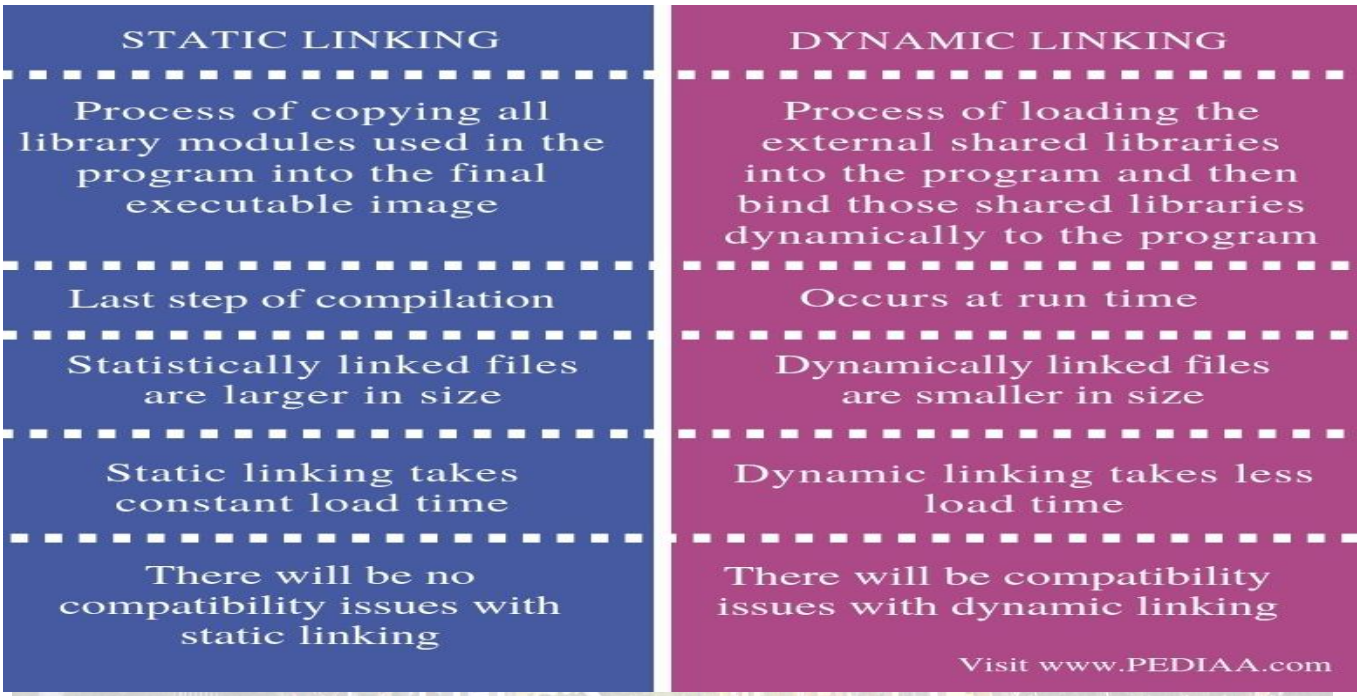
1. Static Linking :

A static linker takes object files produced by the compiler including library functions and produces and executable file. The executable file contains a copy of every subroutine (user defined or library function.) The biggest disadvantage of the static linking is that each executable file contains its own copy of the library routines. If many programs containing same library routines are executed then memory is wasted.

2. Dynamic Linking :

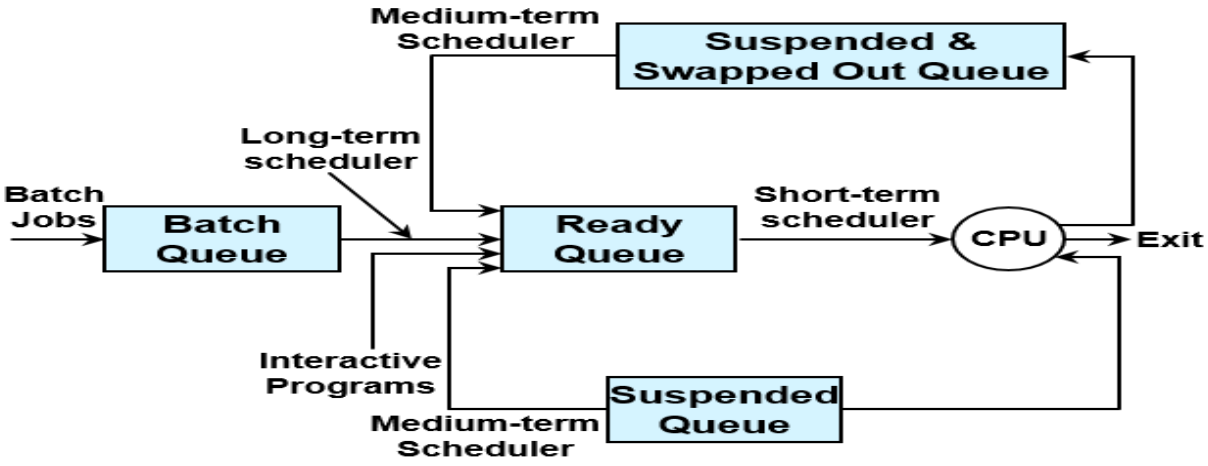
Dynamic linking defers much of the linking process until a program starts running. Dynamic linking involves the following steps:

1) A reference to an external module during run time causes the loader to find the target module and load it.
2) Perform relocation during run time
Dynamic linking permits a program to load and unload routines at run time.



Q 3. a) Explain following Types of Schedulers.
1. Short term
2. Long term
3. Medium term **9-Marks**
Ans :-

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategies. Process scheduling is an essential part of a Multiprogramming operating systems.



1) Long Term Scheduler

It selects the process that are to be placed in ready queue. The long term scheduler basically decides the priority in which processes must be placed in main memory. Processes of long term scheduler are placed in the ready state because in this state the process is ready to execute waiting for calls of execution from CPU which takes time that's why this is known as long term scheduler.

2) Mid – Term Scheduler

It places the blocked and suspended processes in the secondary memory of a computer system. The task of moving from main memory to secondary memory is called **swapping out**.The task of moving back a swapped out process from secondary memory to main memory is known as **swapping in**. The swapping of processes is performed to ensure the best utilization of main memory.

3) Short Term Scheduler

It decides the priority in which processes is in the ready queue are allocated the central processing unit (CPU) time for their execution. The short term scheduler is also referred as central processing unit (CPU) scheduler.

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

b) Explain Seven state process model with diagram? Also explain difference between five state process and Seven state process Model 8-Marks

➔ **Seven state process model :**

Seven state process model

The states present in seven state models are as follows −

New − Contains the processes that are newly coming for execution.

Ready − Contains the processes that are present in main memory and available for execution.

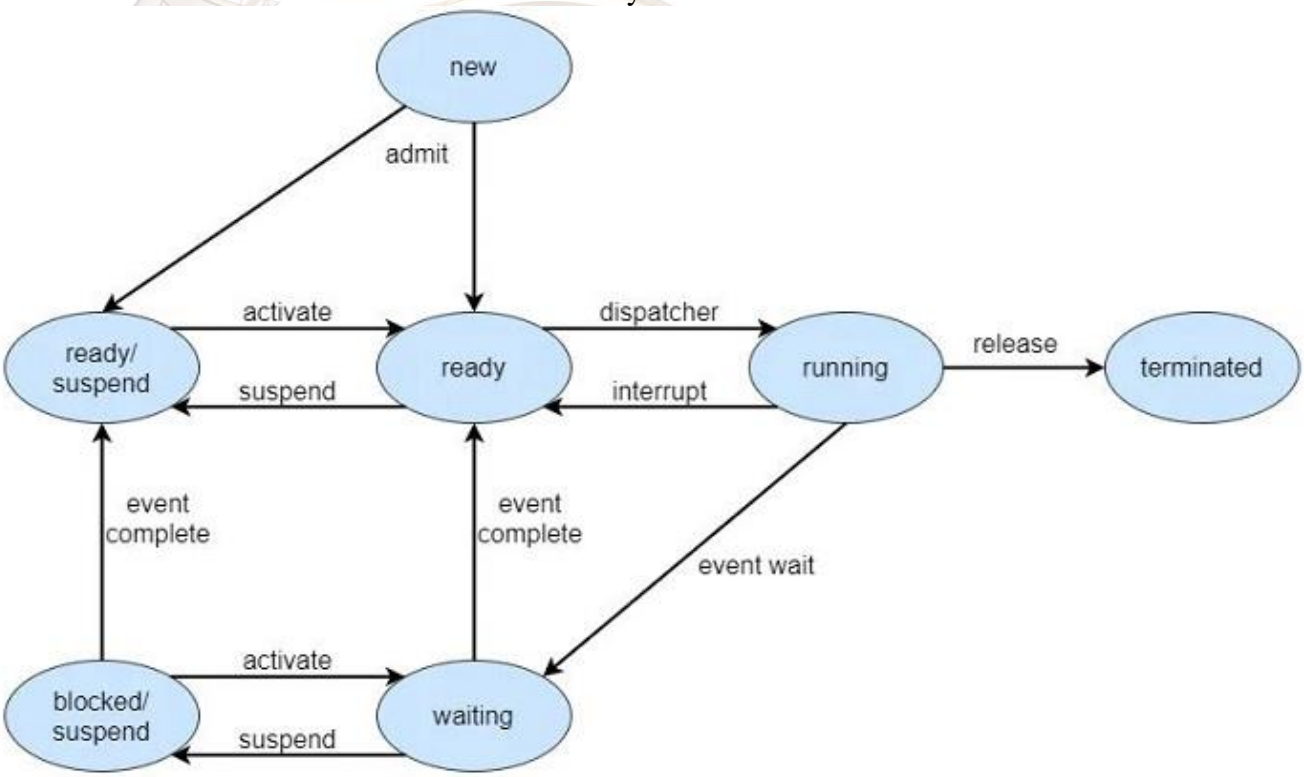
Running − Contains the process that is running or executing.

Exit − Contains the processes that complete its execution.

Blocked − Contains the processes that are present in main memory and awaiting an event to occur.

Blocked Suspend − It contains the process present in secondary memory and awaits an event to occur.

Ready Suspend − Contains the processes that are present in secondary memory but is available for execution as soon as it is loaded into main memory.



Sr. No	Seven State Mode	Sr. No	Five State Model
1	There are seven state process	1	There are five state process
2	Block suspended and ready suspended available	2	Block suspended and ready suspended not available
3	Two state model are present in Seven state process model	3	Two state model are not present in Five state process model
4	It consist of new, ready, running, blocked, Suspended ready and Suspended wait. and exit.	4	It consist of new, ready, running, blocked, and exit.

OR

Q 4. a) Draw Gantt chart and calculate average turnaround time, Average Waiting time for the following processes using SJF non preemptive and round robin with the time quantum 0.5 Unit.

Consider the following set of jobs :

Job No.	Arrival time	Run time
1	10	2
2	10	1
3	11	1
4	12	1

9-Marks

Ans :-

(i) Shortest job first

Gantt chart for scheduling is shown below :

Calculation of turnaround time :

Sr. No.	Process No.	Arrival Time	Finish Time	Turnaround Time
1	P2	10	11	11 – 10 = 1
2	P3	11	12	12 – 11 = 1
3	P4	12	13	13 – 12 = 1
4	P1	10	15	15 – 10 = 5

(iii) Round Robin (Time slab = 0.5)

Finding completion time of each process :

Time	Ready Queue	Next job to be executed
10	P1 P2	P1 for 0.5 time
10.5	P2 P1	P2 for 0.5 time
11	P1 P3 P2	P1 for 0.5 time
11.5	P3 P2 P1	P3 for 0.5 time
12	P2 P1 P4 P3	P2 for 0.5 time P2 finish
12.5	P1 P4 P3	P1 for 0.5 time
13	P4 P3 P1	P4 for 0.5 time
13.5	P3 P1 P4	P3 for 0.5 time P3 finishes
14	P1 P4	P1 for 0.5 time P1 finishes
14.5	P4	P4 for 0.5 time P4 finishes

Calculation of turnaround time

Sr. No.	Job No.	Arrival Time	Finish Time	Turnaround Time
1	P2	10	12.5	2.5
2	P3	11	14	3
3	P1	10	14.5	4.5
4	P4	12	15	3

b) What is meant by threads, Explain thread lifecycle with diagram in details 8-Marks

Ans : There are two types of macro parameters:

1. A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control.
2. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process.
3. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks.
4. Thread is often referred to as a lightweight process.

Life Cycle of Thread:

1. **Born State**: A thread that has just been created.

2. **Ready State:** The thread is waiting for the processor (CPU).

3. **Running:** The System assigns the processor to the thread means that the thread is being executed.

4. **Blocked State:** The thread is waiting for an event to occur or waiting for an I/O device.

5. **Sleep:** A sleeping thread becomes ready after the designated sleep time expires.

6. **Dead:** The execution of the thread is finished.

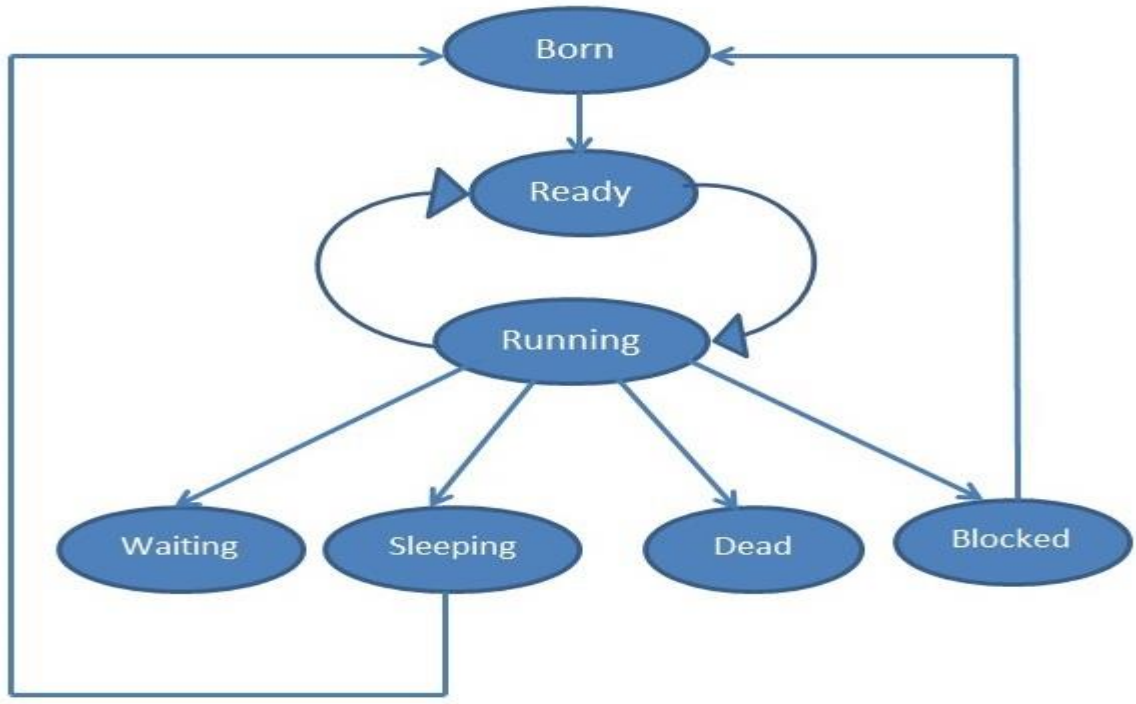


Fig : Life Cycle of a thread

Q. 5 a) Write a short note on following with example? 9 - Marks

1. **Semaphore**
2. **Monitor**
3. **Mutex**

➔ **1. Semaphore :** Semaphore is simply a variable that is non-negative and shared between threads. A semaphore is a signaling mechanism, and a thread that is waiting on a semaphore can be signaled by another thread. It uses two atomic operations, 1) Wait, and 2) Signal for the process synchronization.

A semaphore either allows or disallows access to the resource, which depends on how it is set up.

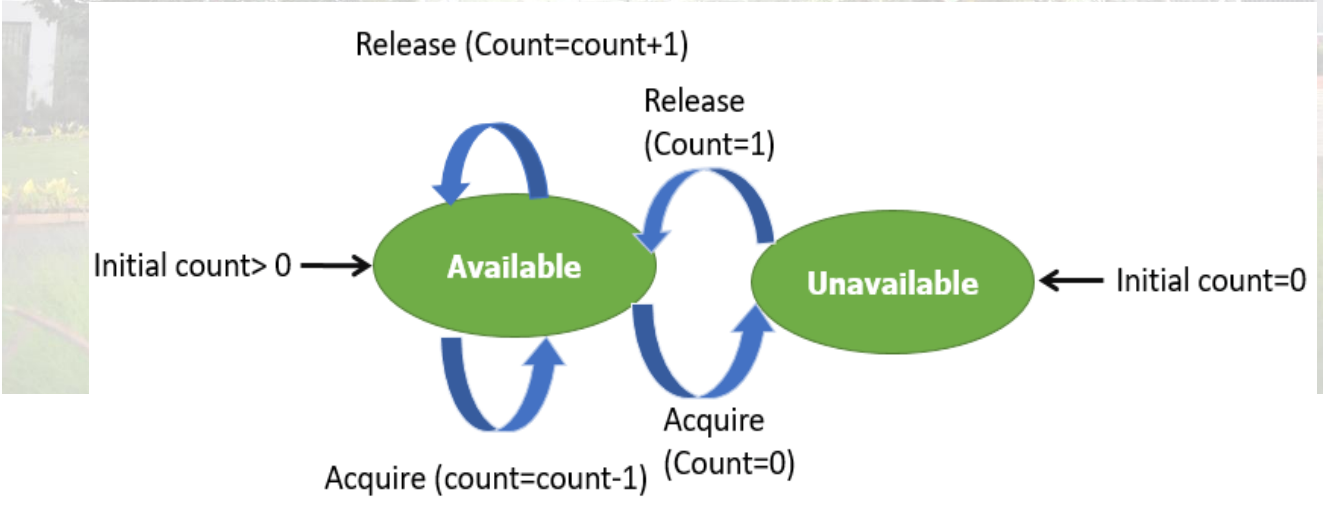
Types of Semaphores

The two common kinds of semaphores are

1. Counting semaphores
2. Binary semaphores.
3. Counting Semaphores

Counting Semaphore :

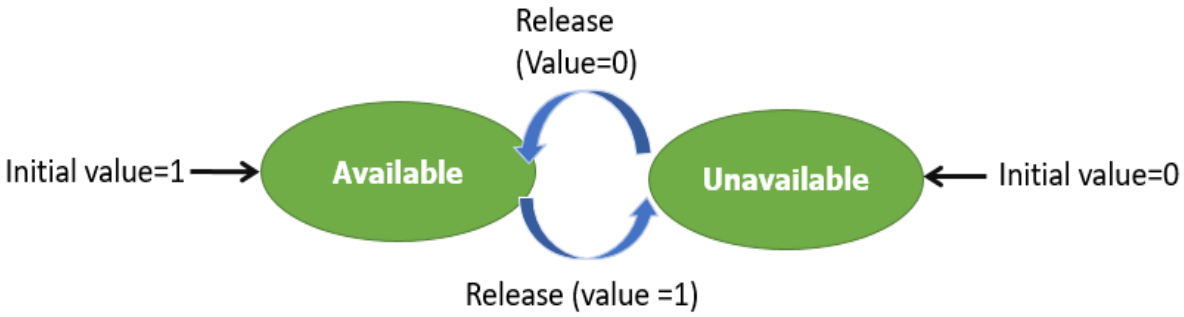
This type of Semaphore uses a count that helps task to be acquired or released numerous times. If the initial count = 0, the counting semaphore should be created in the unavailable state.



However, If the count is > 0, the semaphore is created in the available state, and the number of tokens it has equals to its count.

Binary Semaphores :

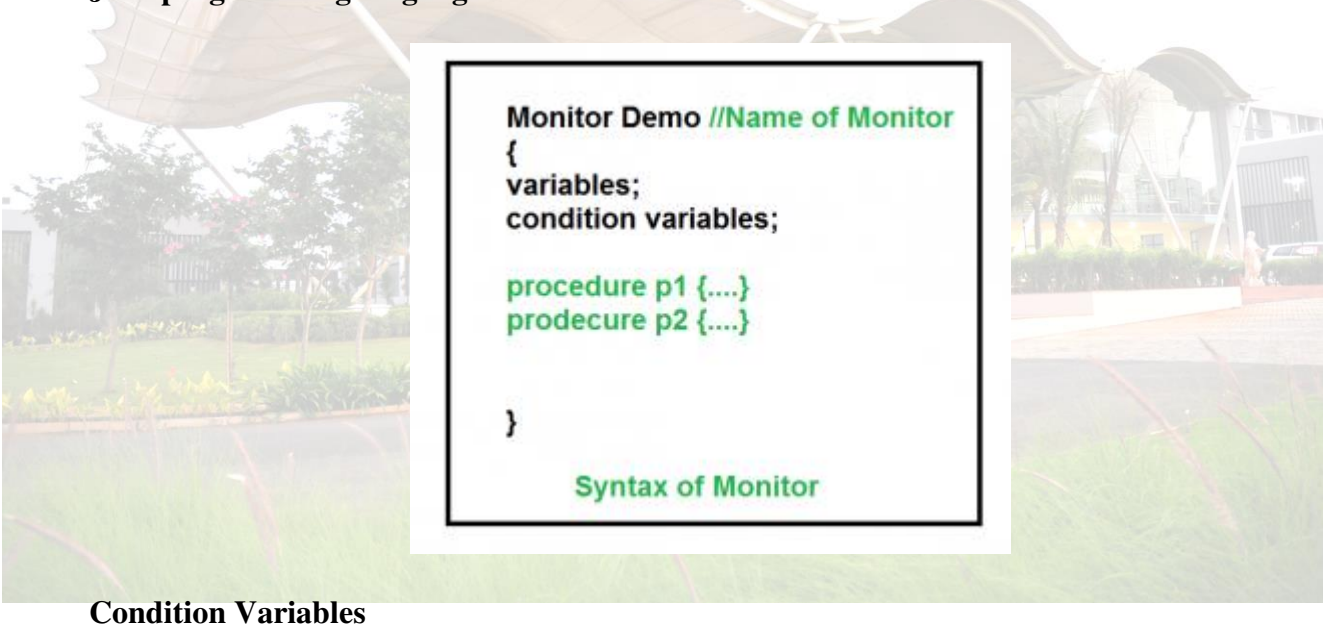
The binary semaphores are quite similar to counting semaphores, but their value is restricted to 0 and 1. In this type of semaphore, the wait operation works only if semaphore = 1, and the signal operation succeeds when semaphore= 0. It is easy to implement than counting semaphores.



2) Monitor :

Monitors are a programming language component that aids in the regulation of shared data access. The Monitor is a package that contains shared data structures, operations, and synchronization between concurrent procedure calls. Therefore, a monitor is also known as a synchronization tool. Java, C#, Visual Basic, Ada, and concurrent Euclid are among some of the languages that allow the use of monitors. Processes operating outside the monitor can't access the monitor's internal variables, but they can call the monitor's procedures.

For example, synchronization methods like the wait() and notify() constructs are available in the **Java programming language.**



Condition Variables

There are two sorts of operations we can perform on the monitor's condition variables:

1. Wait
2. Signal

Consider a condition variable (y) is declared in the monitor:

y.wait(): The activity/process that applies the wait operation on a condition variable will be suspended, and the suspended process is located in the condition variable's block queue.

y.signal(): If an activity/process applies the signal action on the condition variable, then one of the blocked activity/processes in the monitor is given a chance to execute.

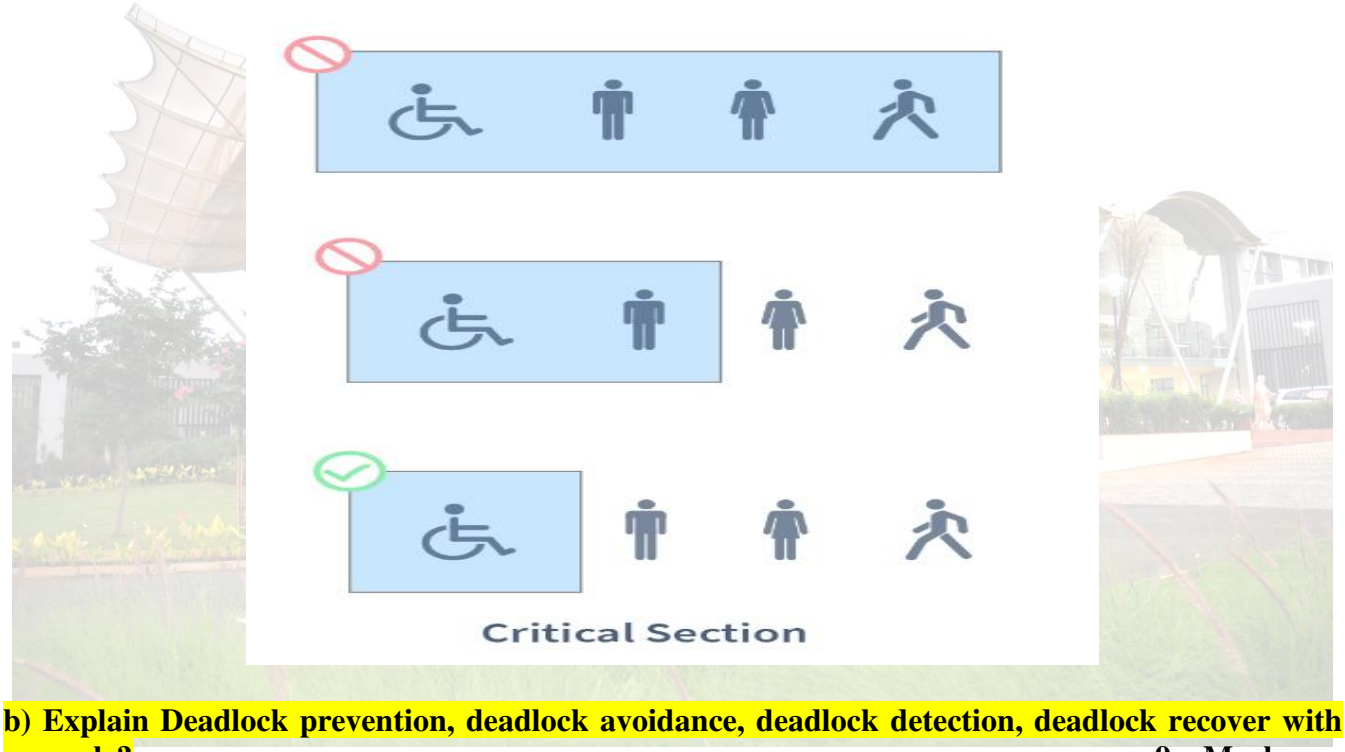
3. **Mutex :** The word "mutex" stands for an object providing MUTual EXclusion between threads. Mutex ensures that only one thread has access to a critical section or data by using operations like a lock and unlock. A thread having the lock of mutex can use the critical section while other threads must wait till the lock is released.

Mutex is special a binary semaphore that synchronizes the access to shared resources like memory or I/O. It is a locking mechanism.

Use of Mutex

In case multiple threads want to access the critical section, mutex allows only one thread at a time to be in the critical section.

Mutex ensures that the code in the critical section (which has shared resources) being controlled will only be used by a single thread at a time.



b) Explain Deadlock prevention, deadlock avoidance, deadlock detection, deadlock recover with example? **9 – Marks**

→ **Deadlock prevention :**

Deadlock prevention is eliminating one of the necessary conditions of deadlock so that only safe requests are made to OS and the possibility of deadlock is excluded before making requests.

As now requests are made carefully, the operating system can grant all requests safely.

Here OS does not need to do any additional tasks as it does in deadlock avoidance by running an algorithm on requests checking for the possibility of deadlock.

Deadlock prevention techniques :

- Mutual Exclusion
- Hold and Wait
- No preemption
- Circular Wait

2. Deadlock avoidance :

In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuously be checked for safe and unsafe states.

In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution.

The simplest and most useful approach states that the process should declare the maximum number of resources of each type it may ever need. The Deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.

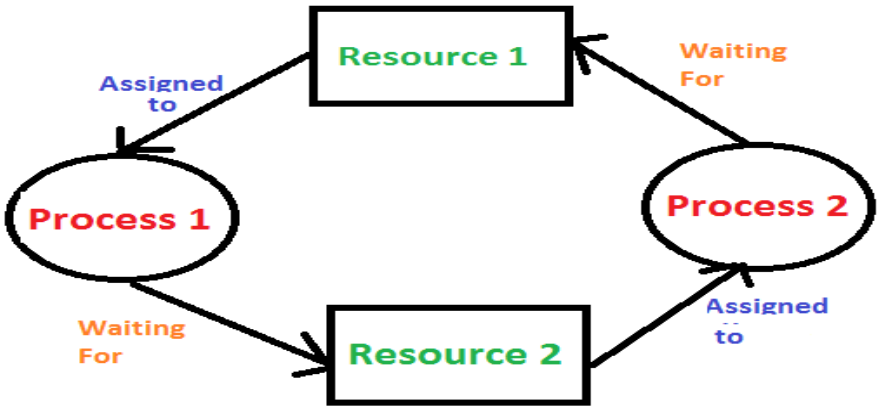
Safe State and Unsafe State

A state is safe if the system can allocate resources to each process(up to its maximum requirement) in some order and still avoid a deadlock. Formally, a system is in a safe state only, if there exists a safe sequence. So a safe state is not a deadlocked state and conversely a deadlocked state is an unsafe state.

In an Unsafe state, the operating system cannot prevent processes from requesting resources in such a way that any deadlock occurs. It is not necessary that all unsafe states are deadlocks; an unsafe state may lead to a deadlock.

3. Deadlock detection :

1. If resources have a single instance –
In this case for Deadlock detection, we can run an algorithm to check for the cycle in the Resource Allocation Graph. The presence of a cycle in the graph is a sufficient condition for deadlock.



In the above diagram, resource 1 and resource 2 have single instances. There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.

2. If there are multiple instances of resources –

Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

4. Deadlock recovery :

A traditional operating system such as Windows doesn't deal with deadlock recovery as it is a time and space-consuming process. Real-time operating systems use Deadlock recovery.

Killing the process –

Killing all the processes involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till the system recovers from deadlock. Killing all the processes one by one helps a system to break circular wait condition.

Resource Preemption –

Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

OR

Q. 6

a) Explain Producer Consumer Problem & Dining Philosopher Problem with Solution? 9 – Mark

→ **Producer Consumer Problem :**

The Producer-Consumer problem is a classical multi-process synchronization problem, that is we are trying to achieve synchronization between more than one process.

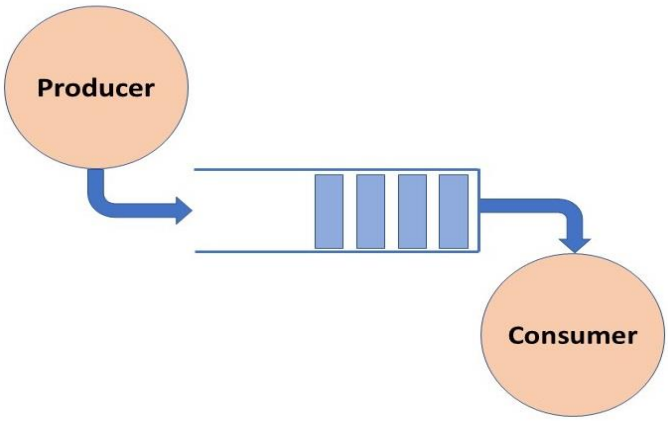
There is one Producer in the producer-consumer problem, Producer is producing some items, whereas there is one Consumer that is consuming the items produced by the Producer. The same memory buffer is shared by both producers and consumers which is of fixed-size.

The task of the Producer is to produce the item, put it into the memory buffer, and again start producing items. Whereas the task of the Consumer is to consume the item from the memory buffer.

Below are a few points that considered as the problems occur in Producer-Consumer:

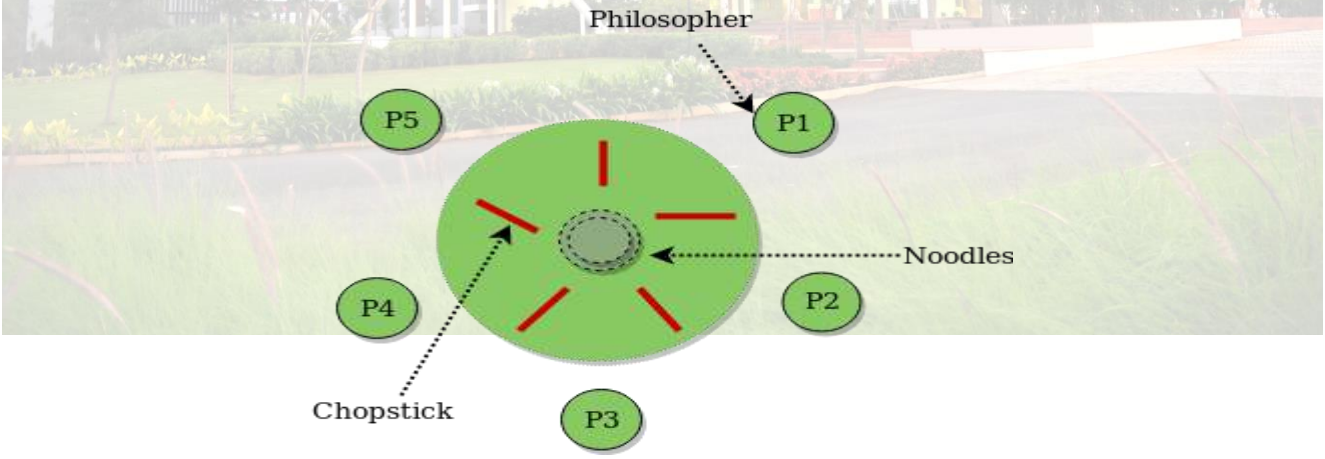
- The producer should produce data only when the buffer is not full. In case it is found that the buffer is full, the producer is not allowed to store any data into the memory buffer.

- Data can only be consumed by the consumer if and only if the memory buffer is not empty. In case it is found that the buffer is empty, the consumer is not allowed to use any data from the memory buffer.
- Accessing memory buffer should not be allowed to producer and consumer at the same time.



Dining Philosopher Problem :

The Dining Philosopher Problem – The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both.



b) What is Deadlock? State and explain the conditions for deadlock, Explain them with example? 9-Mark

→ A deadlock in OS is a situation in which more than one process is blocked because it is holding a resource and also requires some resource that is acquired by some other process. The four necessary conditions for a deadlock situation to occur are mutual exclusion, hold and wait, no preemption and circular set.

Neccessary Conditions for Deadlock

The four necessary conditions for a deadlock to arise are as follows.

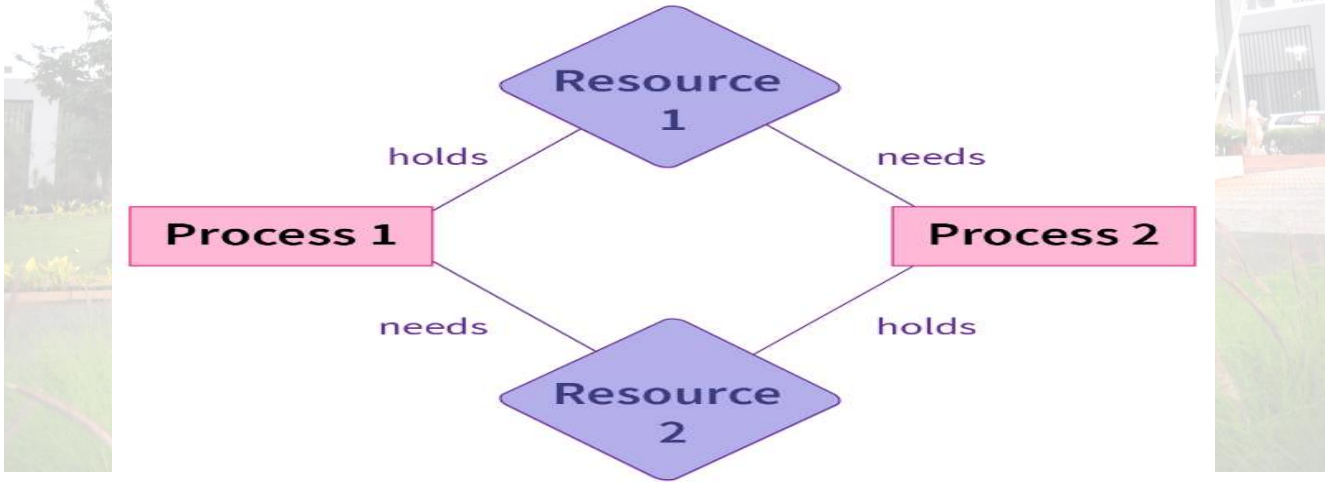
Mutual Exclusion: Only one process can use a resource at any given time i.e. the resources are non-sharable.

Hold and wait: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.

No preemption: The resource can be released by a process voluntarily i.e. after execution of the process.

Circular Wait: A set of processes are waiting for each other in a circular fashion. For example, lets say there are a set of processes {P0, P1, P2, P3} such that P0 depends on P1, P1 depends on P2, P2 depend on P3 and P3 depends on P0. This creates a circular relation between all these processes and they have to wait forever to be executed.

Example :



In the above figure, there are two processes and two resources. Process 1 holds "Resource 1" and needs "Resource 2" while Process 2 holds "Resource 2" and requires "Resource 1". This creates a situation of deadlock because none of the two processes can be executed. Since the resources are non-shareable they can only be used by one process at a time(Mutual Exclusion). Each process is holding a resource and waiting for the other process the release the resource it requires. None of the two processes releases their resources before their execution and this creates a circular wait. Therefore, all four conditions are satisfied.

Q. 7 a) Consider page sequence 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 and discuss working of page replacement policies. Also count page fault (use no. of frame = 3)

1) FIFO 2) LRU 8 – Mark

→ FIFO :

2	**3**	**2**	**1**	**5**	**2**	**4**	**5**	**3**	**2**	**5**	**2**
			1	1	1	4		4		4	2
	3		3	3	2	2		2		5	5
2	2		2	5	5	5		3		3	3
Miss	**Miss**	**Hit**	**Miss**	**Miss**	**Miss**	**Miss**	**Hit**	**Miss**	**Hit**	**Miss**	**Miss**

Total Page Fault are – 09

LRU :

2	**3**	**2**	**1**	**5**	**2**	**4**	**5**	**3**	**2**	**5**	**2**
			1	1		4		4	2		
	3		3	5		5		5	5		
2	2		2	2		2		3	3		
Miss	**Miss**	**Hit**	**Miss**	**Miss**	**Hit**	**Miss**	**Hit**	**Miss**	**Miss**	**Hit**	**Hit**

Total Page Fault are – 07

b) What is meant by fragmentation. Explain Buddy System Fragmentation in details. 9 – Mark

→

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused. It is also necessary to understand that as programs are loaded and deleted from memory, they generate free space or a hole in the memory. These small blocks cannot be allotted to new arriving processes, resulting in inefficient memory use. The conditions of fragmentation depend on the memory allocation system. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to incoming processes**. It is called fragmentation.**

Buddy System Fragmentation :

Static partition schemes suffer from the limitation of having the fixed number of active processes and the usage of space may also not be optimal. The buddy system is a memory allocation and management algorithm that manages memory in power of two increments. Assume the memory size is 2U, suppose a size of S is required.

If 2U-1<S<=2U: Allocate the whole block

Else: Recursively divide the block equally and test the condition at each time, when it satisfies, alloacate the block and get out the loop.System also keep the record of all the unallocated blocks each and can merge these different size blocks to make one big chunk.

Advantage –

1. Easy to implement a buddy system
2. Allocates block of correct size
3. It is easy to merge adjacent holes
4. Fast to allocate memory and de-allocating memory

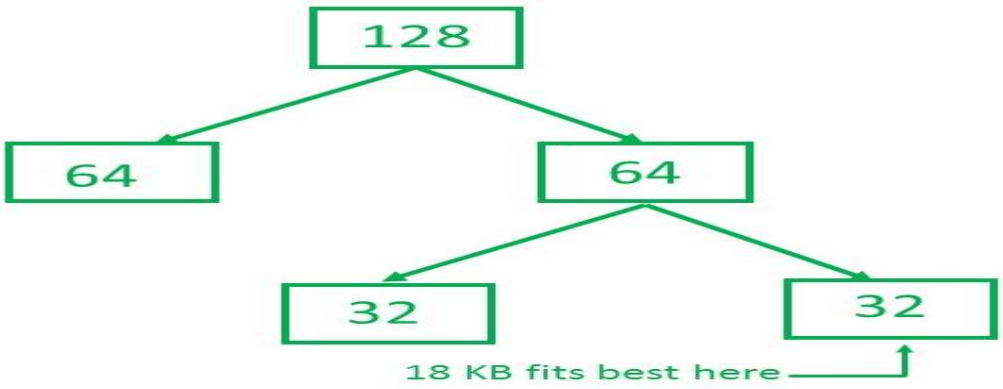
Disadvantage –

1. It requires all allocation unit to be powers of two
2. It leads to internal fragmentation

Example –

Consider a system having buddy system with physical address space 128 KB.Calculate the size of partition for 18 KB process.

Solution –



So, size of partition for 18 KB process = 32 KB. It divides by 2, till possible to get minimum block to fit 18 KB.

OR

Q. 8 a) Write short note on following with diagram 8 - Mark
1) VM with paging
2) VM with segmentation
→

1. VM with paging :

Virtual Memory is a storage mechanism which offers user an illusion of having a very big main memory. It is done by treating a part of secondary memory as the main memory. In Virtual memory, the user can store processes with a bigger size than the available main memory.

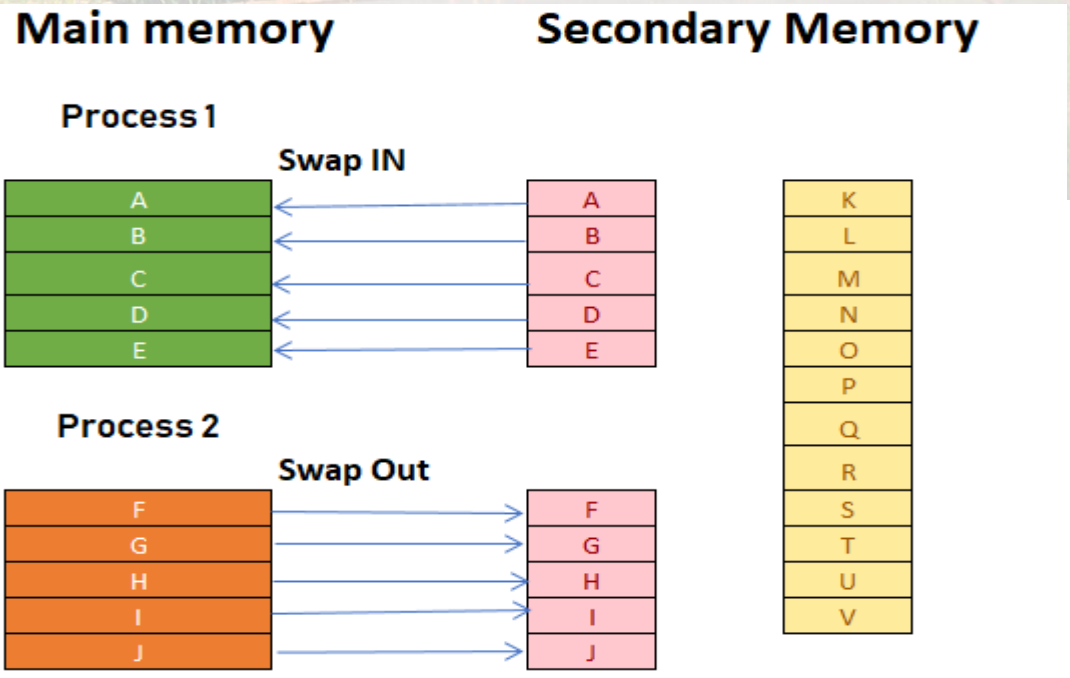
Therefore, instead of loading one long process in the main memory, the OS loads the various parts of more than one process in the main memory. Virtual memory is mostly implemented with demand paging and demand segmentation.

Demand Paging :

A demand paging mechanism is very much similar to a paging system with swapping where processes stored in the secondary memory and pages are loaded only on demand, not in advance.

So, when a context switch occurs, the OS never copy any of the old program's pages from the disk or any of the new program's pages into the main memory. Instead, it will start executing the new program after loading the first page and fetches the program's pages, which are referenced.

During the program execution, if the program references a page that may not be available in the main memory because it was swapped, then the processor considers it as an invalid memory reference. That's because the page fault and transfers send control back from the program to the OS, which demands to store page back into the memory.



2. Virtual Memory With Segmentation :

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the same sizes are called segments. Segmentation gives the user's view of the process which paging does not give. Here the user's view is mapped to physical memory. There are types of segmentation:

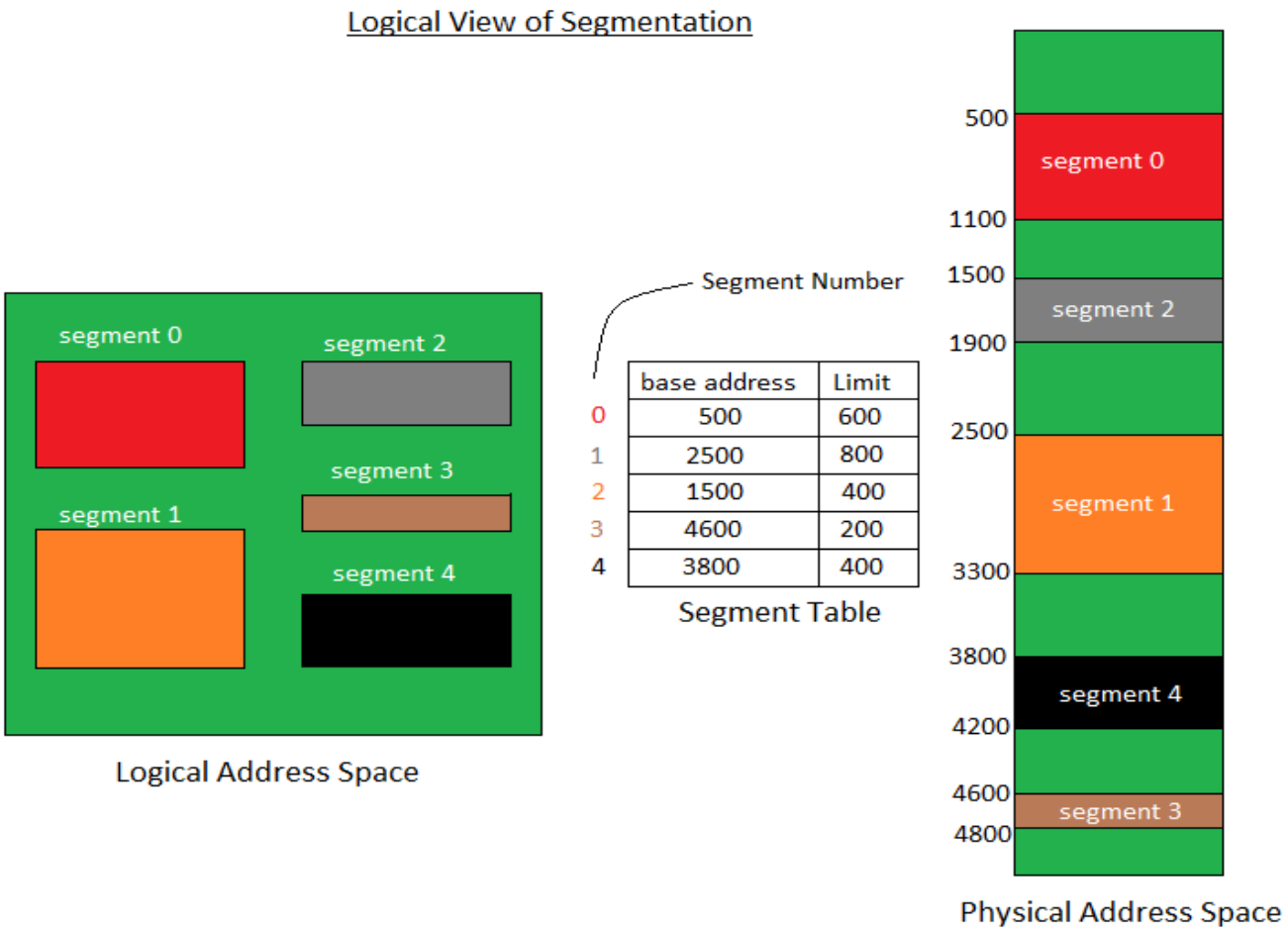
Virtual memory segmentation – Each process is divided into a number of segments, not all of which are resident at any one point in time.

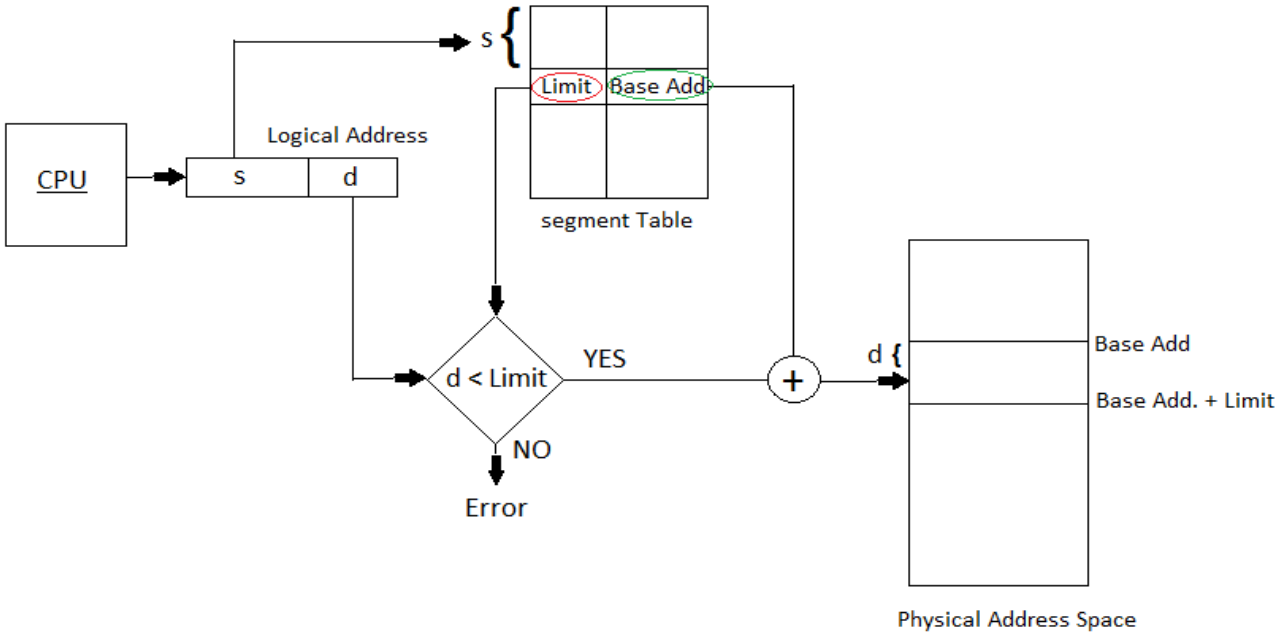
Simple segmentation – Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table. Segment Table – It maps two-dimensional Logical address into one-dimensional Physical address. It's each table entry has:

Base Address: It contains the starting physical address where the segments reside in memory.

Limit: It specifies the length of the segment.



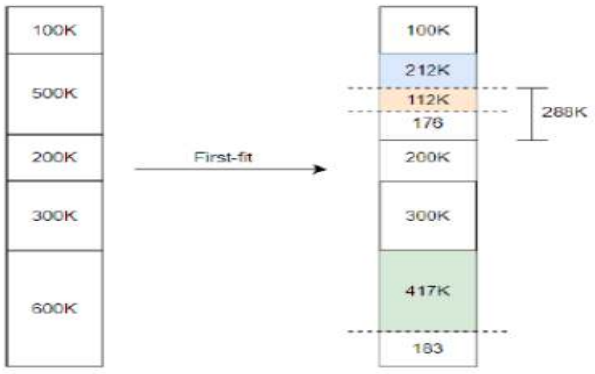


Segment number (s): Number of bits required to represent the segment.

Segment offset (d): Number of bits required to represent the size of the segment.

b) Given the memory partition of size 100K, 500K, 200K, 300K, 600K how would each of the First fit , Best fit and Worst fit algorithm place the processes of 212K , 417K, 426K. Which algorithm makes the most efficient use of memory? 9 – Mark

→ **note : 112K not included in above example :**



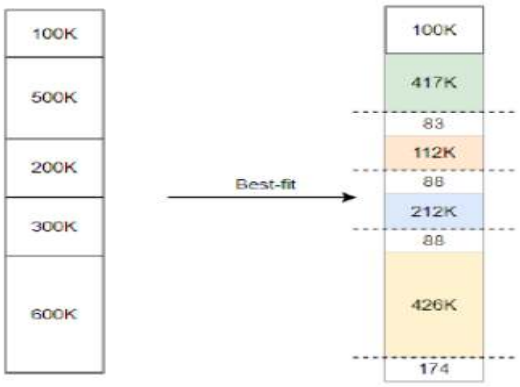
First-fit

212K is put in the 500K partition.

417K is put in the 600K partition.

112K is put in the 288K partition.

426K must wait.



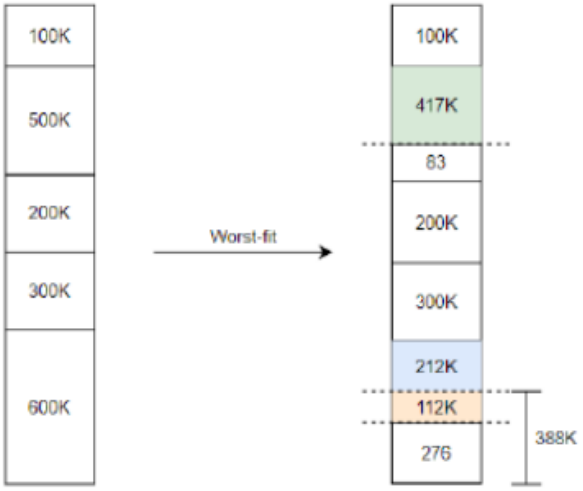
Best-fit

212K is put in the 300K partition.

417K is put in the 500K partition.

112K is put in the 200K partition.

426K is put in the 600K partition.



Worst-fit

212K is put in the 600K partition.

417K is put in the 500K partition.

112K is put in the 388K partition.

426K must wait.

In this case, best-fit makes the most efficient use of memory.

********** THE END **********