Victoria University of Bangladesh
Name: Md. Ziaul Hoque "Sohel"
Student ID: 2221220031
Course Title: Operating System Concept
Course Code: CSI 231
Batch: 22nd (Evening)

Ans to the Que No 1(A)

Valid bit and Invalid bit:

Valid indicates that the associated page is in the logical address space.

Invalid indicates that the associated page is not in logical address space.

Ans to the Que No 1(B)

Architecture of Segmentation:

- Logical address consists of a two tuple: <segment-number, offset>,
- Segment table – maps two-dimensional physical addresses;

each table entry has: base – contains the starting physical address where the segments reside in memory.

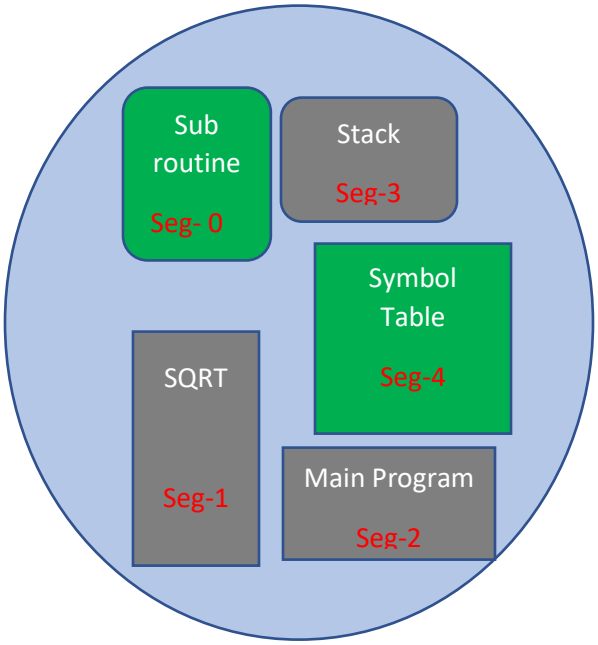
- limit – specifies the length of the segment
- Segment-table base register (STBR) points to the segment table's location in memory.
- Segment-table length register (STLR) indicates number of segments used by a program;

Segment number s is legal if s < STLR

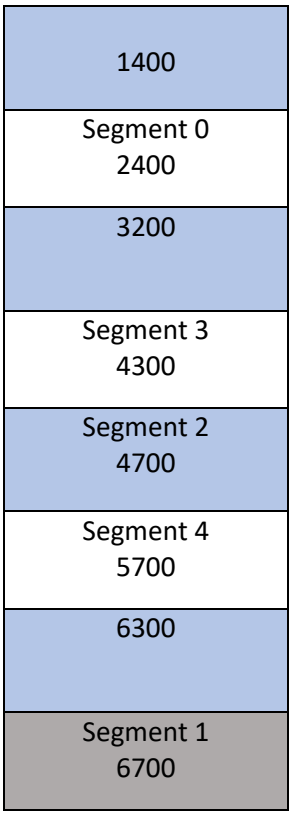
- Protection
- With each entry in segment table associate:
- validation bit = 0
- illegal segment
- read/write/execute privileges
- Protection bits associated with segments; code sharing occurs at segment level.
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem.

A segmentation example is shown in the following

Diagram :



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



Logical Address Space Memory

Segment table

Physical

Ans to the Que No 1(C)

Physical address:

a physical address refers to either a memory location, identified in the form of a binary number, or a media access control (MAC) address. A physical address is also known as a binary address or a real address. Physical address is that address which is seen by the memory unit.

Ans to the Que No 2(A)

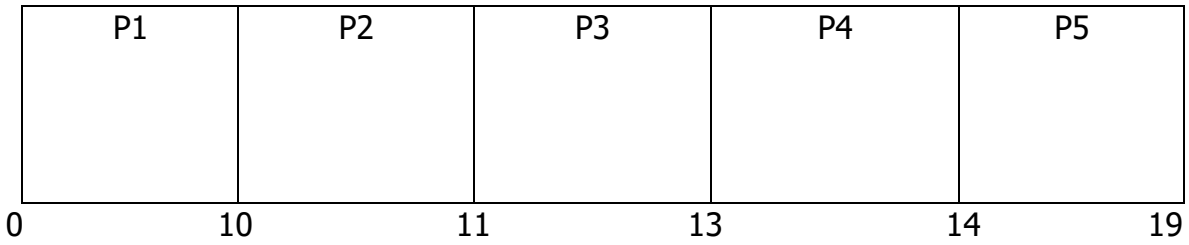
Given set of Processes with the length of CPU burst time given in

Milliseconds:

Process	Arrival Time	Burst Time	Priority
P1	0	10	3
P2	1	1	1
P3	2	2	4
P4	3	1	5
P5	4	5	2

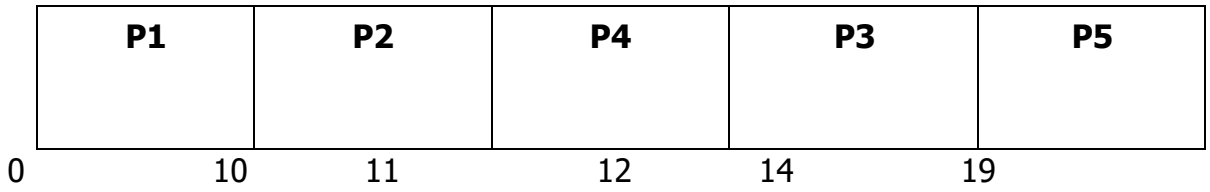
1. FCFS
2. SJF- Non-Preemptive
3. Priority- Preemptive
4. Round Robin with time Quantum 2 & 4

1. FCFS Gantt Chart



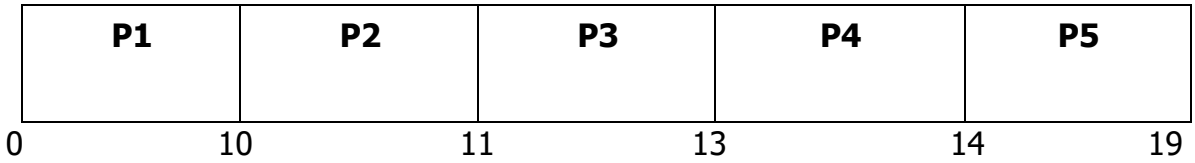
Process	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	10	10	10	0
P2	1	1	11	10	9
P3	2	2	13	11	9
P4	3	1	14	11	10
P5	4	5	19	15	10
			Average	57/5=11.4 ms	38/5=7.6 ms

1. SJF- Non Preemptive Gantt Chart



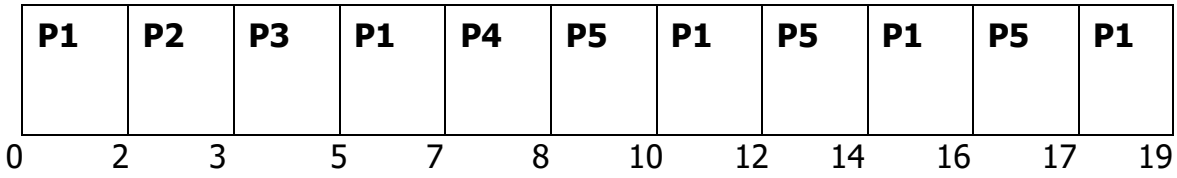
Process	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	10	10	10	0
P2	1	1	11	10	9
P3	2	2	14	12	10
P4	3	1	12	9	8
P5	4	5	19	15	10
			Average	56/5=11.2ms	37/5=7.4ms

2. Priority Preemptive Gantt Chart



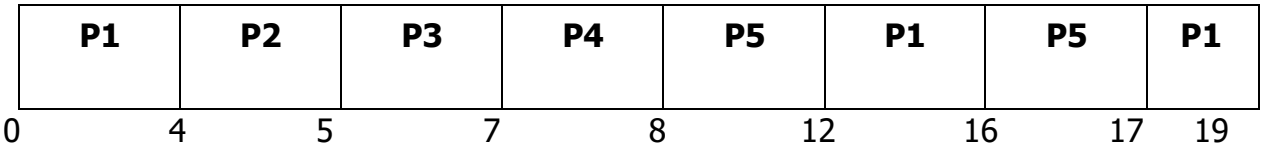
Process	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	10	10	10	0
P2	1	1	11	10	9
P3	2	2	13	11	9
P4	3	1	14	11	10
P5	4	5	19	15	10
			Average	57/5=11.4ms	38/5= 7.6ms

3. Round Robin Gantt Chart (Time Quantum 2)



Process	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	10	19	19	9
P2	1	1	3	2	1
P3	2	2	5	3	1
P4	3	1	8	5	4
P5	4	5	17	13	8
			Average	42/5=8.4 ms	23/5=4.6 ms

4. Round Robin Gantt Chart (Time Quantum 4)



Process	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
P1	0	10	19	19	9
P2	1	1	5	4	3
P3	2	2	7	5	3
P4	3	1	8	5	4
P5	4	5	17	13	8
			Average	46/5=9.2 ms	27/5=5.4ms

Ans to the Que No 2(B)

Multilevel feedback queue:

A multilevel feedback queue is a scheduling algorithm. Scheduling algorithms are designed to have some process running at all times to keep the central processing unit (CPU) busy. The multilevel feedback queue extends standard algorithms with the following design requirements:

1. Separate processes into multiple ready queues based on their need for the processor.
2. Give preference to processes with short CPU bursts.
3. Give preference to processes with high I/O bursts.

Ans to the Que No 3(A)

Characterization of Deadlock:

When all four following requirements are true at once, deadlock can result-

- Mutual exclusion: only one process at a time can use a resource.
- Hold and wait: a process holding at least one resource is waiting to acquire additional resources held by other processes.
- No preemption: a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- Circular wait: there exists a set {P0, P1, ..., P0} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, ..., Pn–1 is waiting for a resource that is held by Pn, and P0 is waiting for a resource that is held by P0.

Ans to the Que No 3(B)

Steps to Deadlock Prevention:

- **Mutual Exclusion –** not required for sharable resources; must hold for non-sharable resources.
- **Hold and Wait** – must guarantee that whenever a process requests a resource, it does not hold any other resources.
  - ✓ Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none.
  - ✓ Low resource utilization; starvation possible.

- **No Preemption –**
  - ✓ If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released.
  - ✓ Preempted resources are added to the list of resources for which the process is waiting.
  - ✓ Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.

- **Circular Wait –** impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

Ans to the Que No 3(C)

Safe State:

When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state.

System is in safe state if there exists a safe sequence of all processes.

- Sequence <P1, P2, ..., Pn> is safe if for each Pi, the resources that Pi can still request can be satisfied by currently available resources + resources held by all the Pj, with j<I.

- If Pi resource needs are not immediately available, then Pi can wait until all Pj have finished.

- When Pj is finished, Pi can obtain needed resources, execute, return allocated resources, and terminate.

- When Pi terminates, Pi+1 can obtain its needed resources, and so on.

- If a system is in safe state →→ no deadlocks.
- If a system is in unsafe state →→ possibility of deadlock.
- Avoidance→→ensure that a system will never enter an unsafe state.

Ans to the Que No 5(A)

Multi-threading Issues:

There are a few multi-threading-related problems below. The old proverb "All good things, come at a price" applies here.

- **Thread Cancellation:**

Terminating a thread before it has finished working is referred to as thread cancellation. There are two possible strategies for this. The first is asynchronous cancellation, which immediately kills the target thread. The other is deferred cancellation, which enables the target thread to check for cancellation on a regular basis.

- **Signal Handling:**

In UNIX systems, signals are used to inform a process that a specific event has taken place. Which thread must a signal be given to when a multi-threaded process gets one? It can be sent to everyone or single thread.

- **Fork System Call:**

A process can make a replica of itself by using the system function fork, which is carried out in the kernel. The issue with multi-threaded processes right now is whether or not the entire process will be replicated if one thread forks.

- **Security Issues:**

Yes, there may be security concerns due to the substantial resource sharing between numerous threads.

Ans to the Que No 5(B)

Semaphore:

A semaphore is an integer variable, shared among multiple processes. The main aim of using a semaphore is process synchronization and access control for a common resource in a concurrent environment.

Properties of Semaphore:

- It's simple and always have a non-negative Integer value.
- Works with many processes.
- Can have many different critical sections with different semaphores.
- Each critical section has unique access semaphores.
- Can permit multiple processes into the critical section at once, if desirable.

Ans to the Que No 5(C)

Worst Fit:

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