Printed Pages - Roll No

…….

***Guru Ghasidas* Vishwavidyalaya (Bilaspur)**
BSC(Third Semester)
Examination, 2014
Introduction to Operating System
Paper Code: PCSC-301

Model Answer

1. Define medium term scheduler.
   Medium term scheduling is part of the swapping function. It removes the processes from the memory. It reduces degree of multiprogramming. The medium term scheduler is in charge of handling the swapped out-processes.

2. What is soft real-time system?
   The Priority scheduling, with no degradation,Low dispatch latency, Preemptible system calls,No virtual memory (or allow pages to be locked).Linux: guarantees about relative timing of tasks, no guarantees about syscalls .

3. What is Dispatcher?
   Another component that is 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. This function involves the following:
   - Switching context
   - Switching to user mode
   - Jumping to the proper location in the user program to restart that program.

4. What is logical address space?
   Set of logical address or collection of logical address generated by cpu.

5. What is DOS?
   In the 1980s or early 1990s, the *operating system* that shipped with most PCs was a version of the *Disk Operating System (DOS)* created by Microsoft.
6. What is MMU?
   Hardware device that maps virtual to physical address. In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.The user program deals with logical addresses; it never sees the real physical addresses.

7. Define turnaround time.
   Turnaround Time: mean time from submission to completion of process.

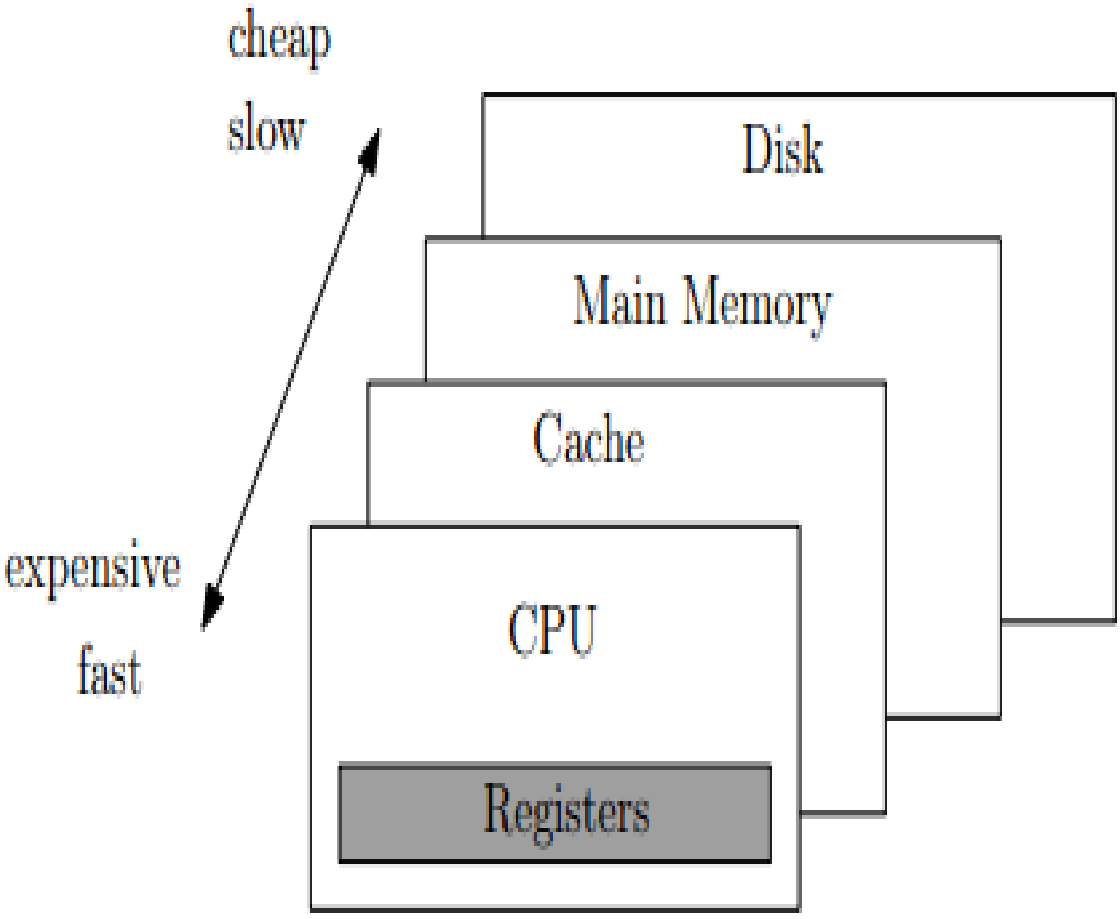
8. What is semaphore?
   A semaphore is a protected variable whose value can be accessed and altered only by the operations P and V and initialization operation called 'Semaphoiinitislize'. Binary Semaphores can assume only the value 0 or the value 1 counting semaphores also called general semaphores can assume only nonnegative values.

9. Explain primary objective of Operating System.
   The operating system is the layer between the hardware and the programs you run. It gives programs a standard interface to the hardware, otherwise every program would need to include its own device drivers. It keeps track of the file system and provides security measures to protect data.

10. Define multiprogramming.
    It is capable of **multiprogramming**, or executing many programs concurrently. It makes multiprogramming possible by capturing and saving all the relevant information about the interrupted program before allowing another program to execute.

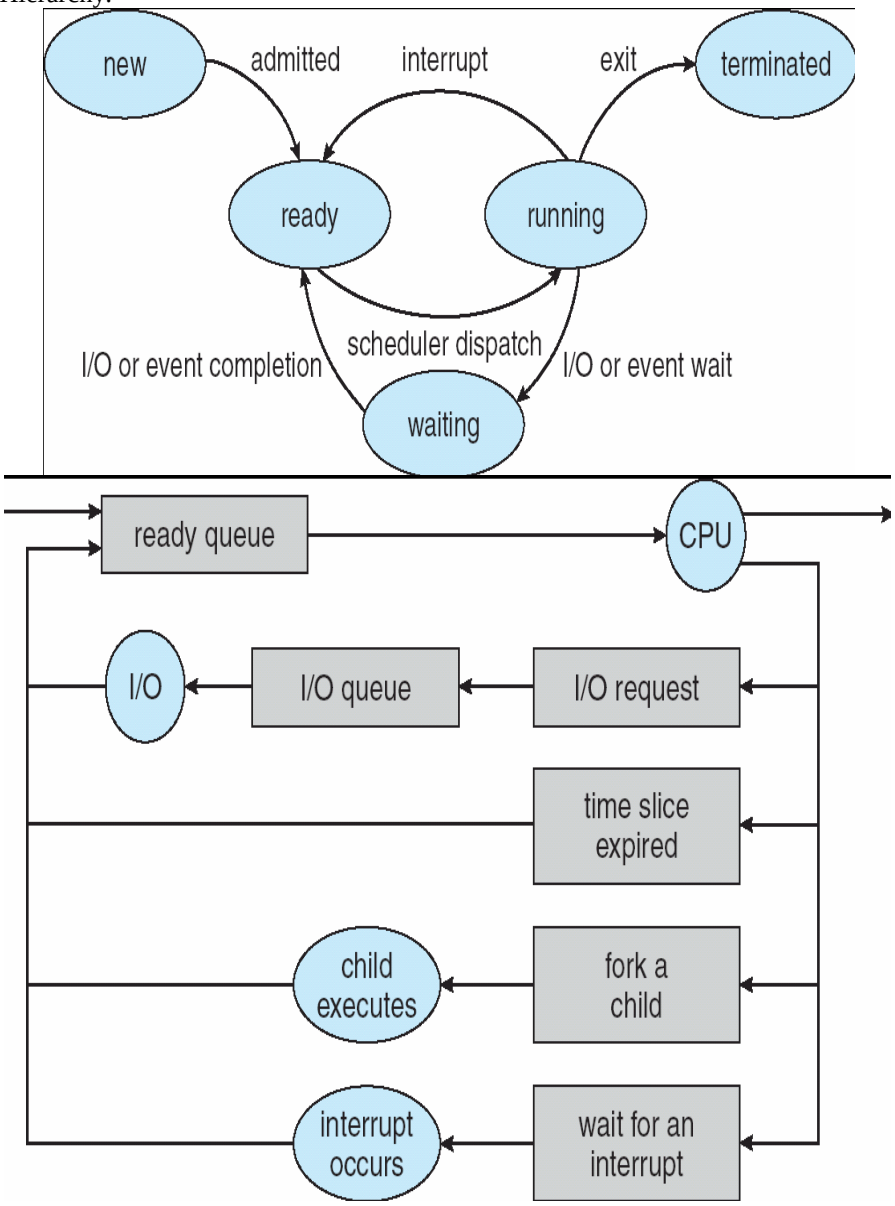
Section B

2 Explain Memory Hierarchy with diagram.



Please describe in details.

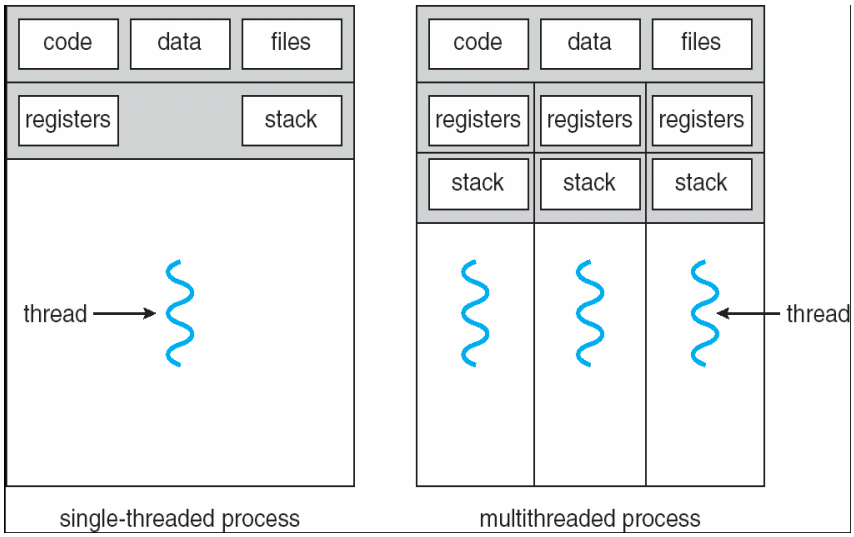
3 Explain Process Hierarchy.



Please describe in details.

4 What is thread and types of thread?

A thread is a flow of execution through the process code, with its own program counter, system registers and stack .Threads are a popular way to improve application performance through parallelism. A thread is sometimes called **a light weight process.** Threads represent a software approach to improving performance of operating system by reducing the over head thread is equivalent to a classical process. Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control.



Threads is implemented in two ways :
1. User Level
2. Kernel Level

1 User Level Thread

In a user thread, all of the work of thread management is done by the application and the kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application begins with a single thread and begins running in that thread.

2 Kernel Level Threads

In Kernel level thread, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process. The Kernel maintains context information for the process as a whole and for individuals threads within the process.

5 What is Banker algorithm with example?

Banker's Algorithm

The resource-allocation graph algorithm is not applicable to a resource-allocation system with multiple instances of each resource type. The deadlock-avoidance algorithm that we describe next is applicable to such a system, but is less efficient than the resource-allocation graph scheme. This algorithm is commonly known as the banker's algorithm.

Let n = number of processes, and m = number of resources types.

- ***Available***: Vector of length m. If available [j] = k, there are k instances of resource type R_j available.
- ***Max***: n x m matrix. If Max [i,j] = k, then process P_i may request at most k instances of resource type R_j.
- ***Allocation***: n x m matrix. If Allocation[i,j] = k then P_i is currently allocated k instances of R_j.
- ***Need***: n x m matrix. If $Need$[i,j] = k, then P_i may need k more instances of R_j to complete its task.

$$Need\ [i,j] = Max[i,j] - Allocation\ [i,j].$$

- Check that Request ≤ Available (that is, (1,0,2) ≤ (3,3,2) ⇒ true.

	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 1	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- Executing safety algorithm shows that sequence < P_1, P_3, P_4, P_0, P_2> satisfies safety requirement.
- Can request for (3,3,0) by P_4 be granted?
- Can request for (0,2,0) by P_0 be granted?

6 What is critical section problem and solution for critical section problem?
Explain following.
i. Physical memory.
ii. Virtual memory

A Critical Section is the part of a program that accesses shared resources. Only when a process is in its Critical Section can it be in a position to disrupt other processes. We can avoid race conditions by making sure that no two processes enter their Critical Sections at the same time.

1. Mutual Exclusion - If process P_i is executing in its critical section, then no other processes can be executing in their critical sections
2. Progress - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely
3. Bounded Waiting - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted
   - Assume that each process executes at a nonzero speed
   - No assumption concerning relative speed of the N processes

The physical memory of a computer usually consists of Random Access Memory (RAM) chips and hard drives. RAM is the amount of real storage, and is the total amount of memory installed on a computer.

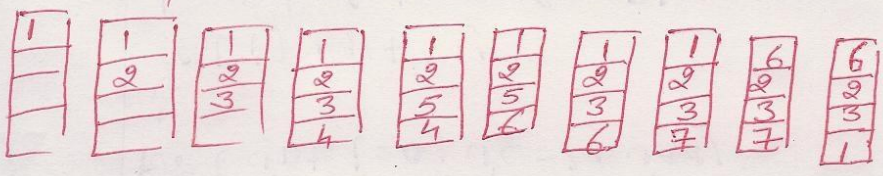
Virtual memory – separation of user logical memory from physical memory. Only part of the program needs to be in memory for execution. Logical address space can therefore be much larger than physical address space. Allows address spaces to be shared by several processes. Allows for more efficient process creation.

Q:-7 Consider the following page reference string:-

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

Sol Page frame = 4

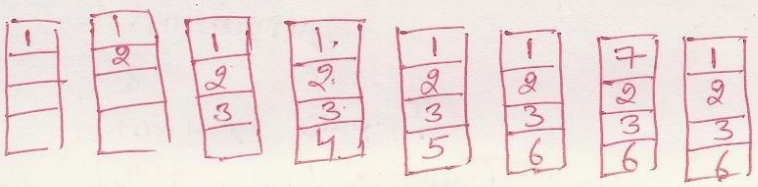
LRU Replacement



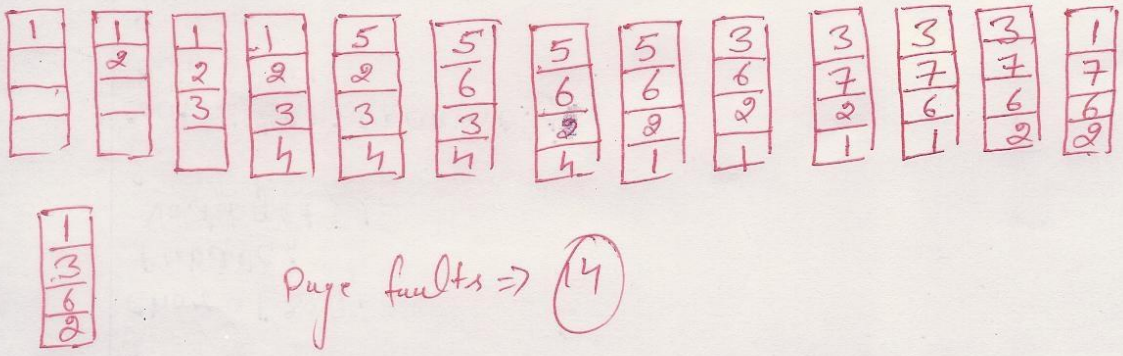
Total Page faults => 10

(iii) optimal Page Replacement Algo Page frame = 4

Page fault :- 8



(ii) FIFO Page Replacement Algo :-

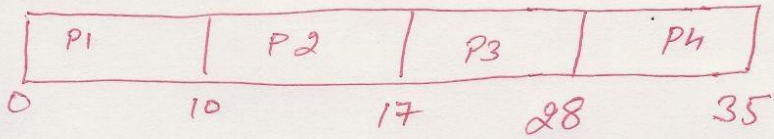


Page faults => (14)

38 Make grantt chart and Find Average using following:-

Process	Arrival Time	Burst Time
P1	0	10
P2	1	7
P3	2	11
P4	3	7

FCFS :-

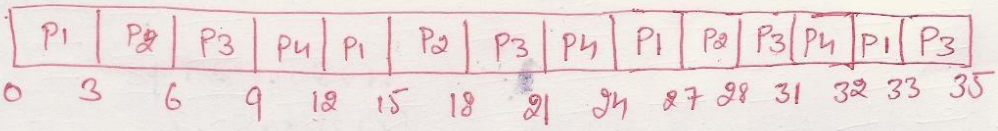


Waiting Time :-

P1 => 0 , P2 => 10, P3 => 17, P4 => 28

Avg w.T. => 0 + 10 + 17 + 28 / 4 => 55/4 => 13.7 ms

Round Roubin Scheduling Algo => Time slice :- 3ms.



W.T. for each Processes :->

P1 -> 0 + (12-3) + (24-15) + (32-27) => 9+9+5 => 23

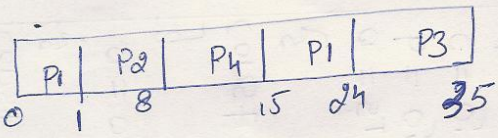
P2 => 3 + (15-6) + (27-18) => 3+9+9 => 21

P3 => 6 + (18-9) + (28-21) + (33-31) => 6+9+7+2 => 24

P4 => 9 + (21-12) + (31-24) => 9+9+7 => 25

Avg w.T. -> 23 + 21 + 24 + 25 / 4 => 23.25 ms

SRF (Shortest Remaining Time First)



W.T

$P_1 \Rightarrow 0 + (15-1) - 0 \Rightarrow 14$

$P_2 \Rightarrow 1 - 1 \Rightarrow 0$

$P_3 \Rightarrow 24 - 2 \Rightarrow 22$

$P_4 \Rightarrow 8 - 3 = 5$

Avg W.T $\Rightarrow$ 14+0+22+5/4

$\Rightarrow$ 41/4 $\Rightarrow$ 10.25 ms.