SPRING MID SEMESTER EXAMINATION-2024

School of Computer Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Operating System
(CS20002/IT 20002/CC20002/CM20002/CS2002)

Time: 1 1/2 Hours **Full Mark: 20**

Answer Any four questions including question No.1 which is compulsory.
The figures in the margin indicate full marks. Candidates are required to give their answers in their own words as far as practicable and all parts of a question should be answered at one place only.

1. Answer all the questions. [1 X 5]

a. Which of the following below statements are correct with respect to user level and kernel level threads:
   i. Context switch is faster in kernel threads
   ii. A system call in user thread can block the entire process
   iii. User threads are transparent to kernel
   iv. All of the above

   Sol: ii, iii are correct.

b. Assume that each process requires 2 seconds of service time in a single-processor system. If new processes are arriving at the rate of 40 processes per two minutes, then calculate the CPU idle rate?

Sol: **Each process need 2 seconds of service time**
There are 40 processes arrived in 2 minutes i.e 120 seconds
Total service time of 40 process are = 40 x 2=80seconds
So CPU idle rate= 40/120=33.33%

c. Take a look at the following statements on process state transition for systems that make use of preemptive scheduling.
   i. A running process can move to ready state.
   ii. A ready process can move to running state.
   iii. A blocked process can move to running state.
   iv. A blocked process can move to ready state.

   Which of the above statement(s) is/are TRUE?

   Sol: i, ii, iv are true.

d. What happens to the process control block (PCB) during a context switch?
   i. It is copied to secondary storage.
   ii. It is swapped out of memory to make space for new processes.
   iii. It is saved and updated with the current state of the process.
   iv. It is deleted from memory to free up resources.

   Sol: iii.

e. A shared variable X is initialized to 5 and three concurrent processes, P, Q and R are executed with the following code:

```
Read(X);
X++;
```

Store(X);

Find out the minimum and maximum values of X after completion of process P, Q, and R.

Sol: Maximum=8, Minimum=6

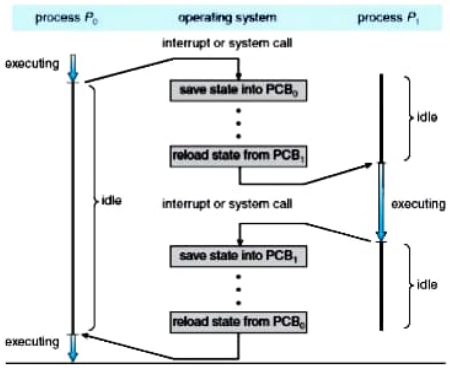
The full marks should be awarded if the solution is fully correct. Otherwise step marks should be awarded for partial correct.

2. [2 x 2.5]

a. What is context switching in OS? Explain the steps involves in context switching of two processes with a neat diagram. What are the triggers that lead to context switches in a system?

Sol: The Context switching is a technique/method used by the OS to switch a process from one state to another to execute its function using CPUs in the system.

When switching perform in the system, it stores the old running process's status in the form of registers and assigns the CPU to a new process to execute its tasks. While a new process is running in the system, the previous process must wait in a ready queue. The execution of the old process starts at that point where another process stopped it. It defines the characteristics of a multitasking operating system in which multiple processes shared the same CPU to perform multiple tasks without the need for additional processors in the system.



Three types of context switching triggers as follows:

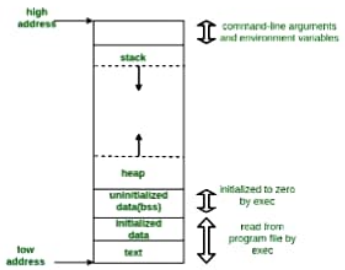
- **Interrupts**
- **Multitasking**
- **Kernel/User switch**

b. Data segment of an address space for a process consist of two parts, called initialized and uninitialized data segments. Describe the need of dividing the data segment further in to two sub parts along with their usability.

Sol: A typical memory representation of a program consists of the different sections as shown in figure: two data segment.

- **Initialized data segment**
- **Uninitialized data segment (bss)**

c.



Initialized Data Segment: Initialized data segment, usually called simply the Data Segment. A data segment is a portion of the virtual address space of a program, which contains the global variables and static variables that are initialized by the programmer.

Uninitialized Data Segment: **Uninitialized data segment often called the "bss" segment, named after an ancient assembler operator that stood for "block started by symbol." Data in this segment is initialized by the kernel to arithmetic 0 before the program starts executing uninitialized data starts at the end of the data segment and contains all global variables and static variables that are initialized to zero or do not have explicit initialization in source code.**

3. As many as 4 processes arrive in the ready queue as given below: [5]

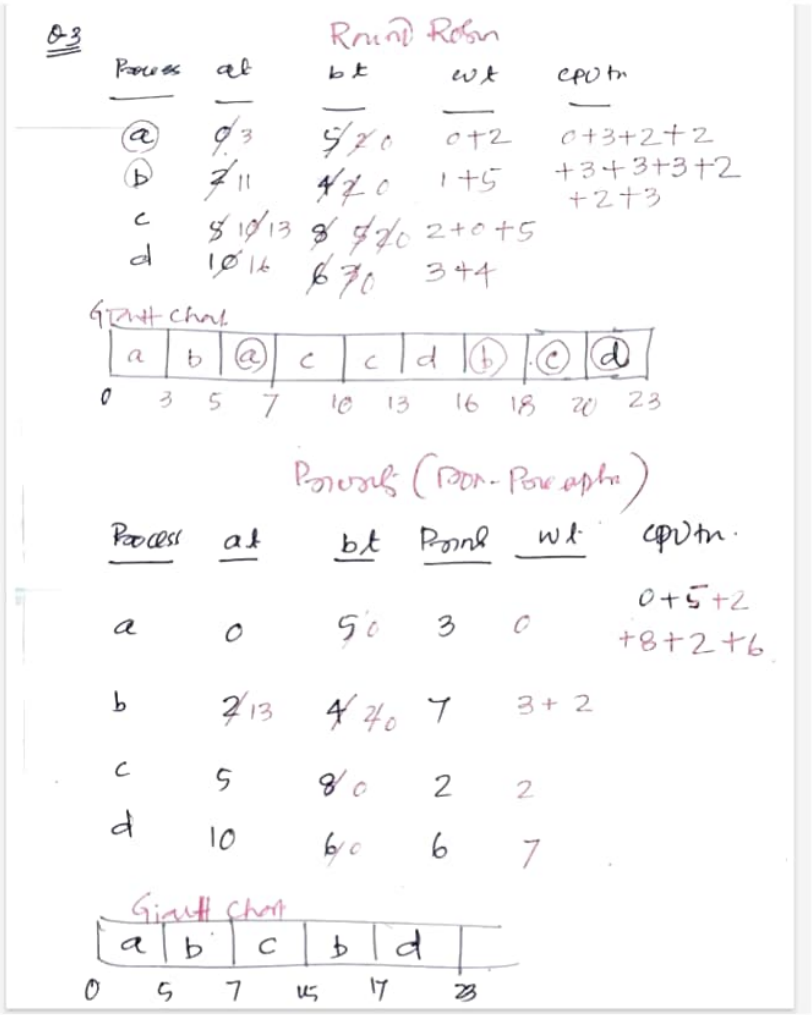
Process Name	Arrival Time in ms	CPU Burst Time in ms	Priority
A	0	5	3
B	2	4	7
C	5	8	2
D	10	6	6

After 2ms of execution, the process B has to wait for 6ms for a resource to continue its execution. Draw the Gantt chart and calculate the waiting time of each process for the following scheduling policies:

i. Round Robin with 3ms time quantum

ii.Non-preemptive Priority with higher digit indicates higher priority .

Sol:



4. [2 x 2.5]

a. Differentiate between multi-programming, multitasking, and multiprocessing and multi-threading.

Sol: Multi-Programming

When the computer runs more than one program in the main memory, it is termed as Multi-programming. e.g. Running Microsoft Word, Paint and Chrome on the processor. Multiple programs runs on a single processor(CPU), with its maximum utilization.

Multi-Tasking

When multi-programming extends to submitting of more than one job or task by the user or multiple users, it is termed as multi-tasking. It is the ability of the operating system to accept more than one job, and it can be either from a single or multiple users. While multi-programming emphasizes sharing the memory by multiple programs, multitasking emphasizes sharing the CPU as well as memory by multiple programs or tasks.

Multi-Processing

When we have more than one processor running to complete a task, it is termed as Multi-Processing. It is the property of the hardware in use. A computer system can have one processor, and termed as uni-processor; or more than one processor and be termed as multi-processor. e.g. A dual core has two processors, whereas quad core has four processors. Its major advantage is that task can run in parallel on multiple processors, thus helping to reduce the computational time.

Multi-Threading

Multi-threading is the ability of a program or an operating system to enable more than one user at a time without requiring multiple copies of the program running on the computer. Multi-threading can also handle multiple requests from the same user.

b. A shared variable 'x' is initialized to zero, is operated by four processes 'W', 'X', 'Y', 'Z'. Processes 'W' and 'X' increment 'x' by one, while processes 'Y' and 'Z' decrement 'x' by two. Each processes before reading perform 'wait' on semaphore's' and 'signal' on 's' after writing . If semaphore's' is initialized to two. Find what the maximum possible value of 'x' after all processes completed execution and explain your answer.

W	X	Y	Z
Wait(s) Read(x) x=x+1 Write(x) Signal(s)	Wait(s) Read(x) x=x+1 Write(x) Signal(s)	Wait(s) Read(x) x=x-2 Write(x) Signal(s)	Wait(s) Read(x) x=x-2 Write(x) Signal(s

Sol: X and W reads x and increment x by 1.

Y and Z reads x and decrement x by 2

Start with X, performs wait(x), so s--, s=1, reads x=0, then x=x+1=1

Then with Y, performs wait(x), so s--, s=0, reads x=0, then x=x-2=-2. Then store x. Signal (s) makes, s=1.

Start with Z, performs wait(x), so s--, s=0, reads x=-2, then x=x-2=-4.Then store x. Signal (s) makes, s=1.

Now X store x, Signal(s) makes s=2. So now x=1.

Now with W, performs wait(x), so s--, s=1, reads x=1, then x=x+1=2. Then store x. Signal (s) makes, s=2. So now x=2.

So maximum is 2.

5. [2 + 3]

a. What is the difference between hardware lock and software lock for critical section? (Do not write down different hardware and software locks.). Determine which critical section requirements are satisfied in the context of two process software solutions as given below.

Process P0	Process P1

turn = 1;
flag[0] = true;
while(flag[1] && turn==1);
critical section
flag[0] = false;
	urn = 0;
	flag[1] = true;
	while(flag[0] && turn==0);
	critical section
	flag[1] = false;

Sol:

Hardware Lock:	**Software Lock:**
➢ **A hardware lock, often referred to as a "mutex", is implemented using hardware support such as atomic instructions or dedicated hardware components like lockable memory locations.** ➢ **Hardware locks are typically more efficient than software locks because they leverage low-level hardware mechanisms for synchronization.** ➢ **They are generally harder to implement and require specialized hardware support, which might limit their portability across different architectures.** ➢ **Hardware locks can ensure mutual exclusion without the need for explicit software code to manage the lock state, which can reduce the potential for programming errors.**	➢ **A software lock, also known as a "mutex" or "semaphore," is implemented entirely in software using programming constructs such as atomic operations, condition variables, or other synchronization primitives provided by the programming language or operating system.** ➢ **Software locks are more portable because they rely solely on software constructs and do not require specific hardware support.** ➢ **They may be less efficient compared to hardware locks because they involve more overhead due to the need for software-level operations and context switches.** ➢ **Software locks are easier to implement and debug because they are implemented using high-level programming constructs that are more familiar to developers.**

Progress and bounded waiting is satisfied.

b. Subsequently, write a proof for each of the critical section requirement, indicating whether it is satisfied or not within the aforementioned two-process solution.

Sol: critical section requirement:

1-<u>Mutual exclusion requiremnet is not satisfied:</u>

Process P0	**Process P1**
1-turn = 1; **6-flag[0] = true;** **7-while(flag[1] && turn==1);** **8-critical section** **flag[0] = false;**	**2-turn = 0;** **3-flag[1] = true;** **4-while(flag[0] && turn==0);** **5-critical section** **flag[1] = false;**

In case of process P0 and P1 are executed in the order of 1->2->3->4->5->6->7->8 as per the line number given above, then while P1 is in CS, still P0 can enter in the CS. Here Mutual exclusion property does not hold good.

2-Progress requirement is satisfied:

P0 cannot enter CS only if stuck in while() with condition flag[1] = true and turn = 1.

Similarly, P1 cannot enter CS only if stuck in while() with condition flag[0] = true and turn = 0.

Case-I:

If P1 is not ready to enter CS then flag[1] = false and P0 can then enter its CS.

Case-II:

If P1 is ready to enter CS then, P1 has set flag[1]=true and is in its while(), then either turn=0 or turn=1.

If turn=0, then P0 enters CS.

If turn=1 then P1 enters CS, will reset flag[1]=false on exit: allowing P0 to enter CS.

3-Bounded waiting requirements is satisfied:

P1 is executing its CS repeatedly, upon exiting its CS, P1 sets flag[1] = false, hence the while loop is false for P0 and it can go to CS.

However, P1 may attempt to re-enter its CS before P0 has a chance to run. But to re-enter, P1 set flag[1]=true and sets turn=0 hence the while loop is true for P1 and it waits. But the while loop is now false for P0 and it can enter into CS. At most one CS entry by P0 (bounded waiting).