******COMSATS University Islamabad (Lahore** **Campus)**

**Assignment 2– SPRING 2022**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Course Title: | Operating System | Course Code: | CSC322 | Credit Hours: | 3+1 |
| Course Instructor: | Kisaa Fatima | Reg # |  | | |
| **Due Date:** | **29-4-2022** | **Maximum Marks:** | | **10** | |
| **Name** | **Saud ul Hassan** | **Regd. Number** | | **SP19-BCS-078** | |
| **Important Instructions / Guidelines:** | | | | | |

# Question No 1.a

# Statement:

Analyze why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems.

# Answer:

Spinlocks are not appropriate for single-processor systems because the condition that would break a process out of the spinlock can be obtained only by executing a different process. If the process is not relinquishing the processor, other processes do not get the opportunity to set the program condition required for the first process to make progress. In a multiprocessor system, other processes execute on other processors and thereby modify the program state in order to release the first process from the spinlock

# Question No 1.b

# Statement:

Examine that, if the wait () and signal () semaphore operations are not executed atomically, then mutual exclusion may be violated.

# Answer:

A wait operation atomically decrements the value associated with a semaphore. If two wait operations are executed on a semaphore when its value is 1, if the two operations are not performed atomically, then it is possible that both operations might proceed to decrement the semaphore value, thereby violating mutual exclusion.

# Question No 2.a

# Statement

Discover the meaning of the term busy waiting. Compare busy waiting with other kinds of waiting in an operating system. Can busy waiting be avoided altogether? Explain your answer.

# Answer:

Busy waiting means that a process is waiting for a condition to be satisfied in a tight loop without relinquishing the processor. Alternatively, a process could wait by relinquishing the processor, and block on a condition and wait to be awakened at some appropriate time in the future. Busy waiting can be avoided but incurs the overhead associated with putting a process to sleep and having to wake it up when the appropriate program state is reached.

# Question No 2.b

# Statement:

Dissect and illustrate how a binary semaphore can be used to implement mutual exclusion among n processes

# Answer:

The n processes share a semaphore, mutex, initialized to 1. Each process Pi is organized as follows:

do {

wait(mutex);

/\* critical section \*/

signal(mutex);

/\* remainder section \*/

} while (true);

wait(mutex){

while(mutex==0); /\*busy waiting\*/

mutex=0;

}

signal(mutex){

mutex = 1;

}