Week 7 – LAQ's

Instructions

Elaborately explain the hardware solution to the process synchronization problem.

Hardware solutions to the process synchronization problem primarily involve using atomic operations that ensure mutual exclusion and prevent race conditions. Key hardware mechanisms include:

Test and Set: This instruction atomically tests a variable and sets it to a new value. If the variable was already set, it indicates that another process is in the critical section. The algorithm works as follows:

A shared variable lock is initialized to false.

The TestAndSet(lock) function returns the current value of lock and sets it to true.

If the first process calls TestAndSet(lock) and it returns false, it enters the critical section.

Other processes will keep calling TestAndSet(lock) until it returns false, ensuring mutual exclusion.

Once the first process exits the critical section, it sets lock back to false, allowing other processes to enter.

Swap: This algorithm is similar to Test and Set but uses a swap operation to achieve mutual exclusion. The process works as follows:

A shared variable lock is used, and a local variable key is initialized to true.

The swap(lock, key) function swaps the values of lock and key.

The first process that successfully swaps lock to true will enter the critical section.

Other processes will keep trying to swap until they find lock is false, ensuring that only one process can be in the critical section at a time.

Unlock and Lock: This algorithm builds on the Test and Set mechanism by adding a waiting array for each process. The steps are:

Each process has a waiting[i] variable to indicate if it is waiting to enter the critical section.

When a process wants to enter, it sets its waiting[i] to true and calls TestAndSet(lock).

If it finds that lock is true, it continues to wait.

When a process exits the critical section, it checks if there are any other processes waiting. If not, it sets lock to false.

Handshaking Protocols: In hardware implementations, handshaking protocols are used to synchronize communication between hardware modules. This involves:

A master-slave relationship where the master initiates a request and the slave acknowledges it.

Two control wires are used: one for the request and another for the acknowledgment.

The master sends a request signal and waits for the acknowledgment before proceeding, ensuring that data is transferred correctly.

Atomic Operations: These are low-level operations that complete in a single step relative to other threads. They are crucial for implementing synchronization primitives like locks and semaphores in hardware.

In summary, hardware solutions to process synchronization problems leverage atomic operations and specific algorithms to ensure that only one process can access shared resources at a time, thus preventing race conditions and ensuring data integrity.