# Operating Systems – Review Questions 2

Group: Ketchup

Author: Ali Sahibi

## 1)

Busy waiting: Blocking thread from making progress while waiting for a lock to become available. This is usually done through keeping the thread stuck in a loop until the lock becomes available, and this is a waste of CPU-resources since the OS will schedule the thread to run on the CPU despite it effectively doing nothing.

Busy waiting can be avoided by suspending the current thread if the lock is unavailable and taken by another thread. When the other thread releases the lock, we will wake up our suspended thread and acquire the lock. While the thread is suspended it is not scheduled to run on the CPU and thus CPU resources are not wasted.

## 2)

An atomic operation guarantees that whenever the operation is called it is executed atomically, meaning that from when it starts executing until it is finished it cannot be interrupted. An atomic operation also guarantees that simultaneous operations on different cores is executed sequentially.

I will now give an example scenario of non-atomic wait() and signal() causing mutual exclusion violation: Assume we have 3 threads: T1, T2, T3 and a semaphore s = 0 at time 0, this is a possible CPU scheduling scenario:

|  |  |  |
| --- | --- | --- |
| Time | CPU1 | CPU2 |
| 0 | T3: Signal(): write s = s + 1 = 1 | T2: Wait(): read s = 0 |
| 1 | T1: Wait(): reads s = 1 | T2:Wait(): read s = 1 |
| 2 | T1: Wait(): write s = s - 1 | T2: Wait(): write s = s - 1 |
| 3 | T1:Access to Critical Section | T2:Access to Critical Section |

At time 0 thread T3 on CPU1 calls signal() and s is incremented. At time 1 both T1 and T2 read s simultaneously and both get the value s = 1. At time 2 both T1 and T2 decrement S. And at time 3 T1 and T2 access the critical section simultaneously violating mutual exclusion. If wait() was an atomic operation, then CPU1 and CPU2 would have performed the wait() operation(both read and write in one instruction) sequentially rather than simultaneously and thus preventing mutual exclusion violation.

## 3)

a.

Need Matrix:

A picture containing text

Description automatically generated

b.

Text, letter

Description automatically generated

Yes, the system is in a safe state with safe thread sequence <T0, T2, T3, T4, T1>.

c.

Yes the since the system is in a safe state after immediate allocation of resources or the request.