LOCK VARIABLE SYNCHRONIZATION MECHANISM

This is the simplest synchronization mechanism. This is a Software Mechanism implemented in User mode. This is a busy waiting solution which can be used for more than two processes.

In this mechanism, a Lock variable lockis used. Two values of lock can be possible, either 0 or 1. Lock value 0 means that the critical section is vacant while the lock value 1 means that it is occupied.

A process which wants to get into the critical section first checks the value of the lock variable. If it is 0 then it sets the value of lock as 1 and enters into the critical section, otherwise it waits.

The pseudo code of the mechanism looks like following.

```
Entry Section →
While (lock! = 0);
Lock = 1;
//Critical Section
Exit Section →
Lock =0;
```

If we look at the Pseudo Code, we find that there are three sections in the code. Entry Section, Critical Section and the exit section.

Initially the value of lock variable is 0. The process which needs to get into the critical section, enters into the entry section and checks the condition provided in the while loop.

The process will wait infinitely until the value of lock is 1 (that is implied by while loop). Since, at the very first time critical section is vacant hence the process will enter the critical section by setting the lock variable as 1.

When the process exits from the critical section, then in the exit section, it reassigns the value of lock as 0.

Every Synchronization mechanism is judged on the basis of four conditions.

1. Mutual Exclusion
2. Progress
3. Bounded Waiting
4. Portability

MUTUAL EXCLUSION

The lock variable mechanism doesn't provide Mutual Exclusion in some of the cases. This can be better described by looking at the pseudo code by the Operating System point of view I.E. Assembly code of the program. Let's convert the Code into the assembly language.

```
Load Lock, R0
CMP R0, #0
JNZ Step 1
Store #1, Lock
Store #0, Lock
```

Let us consider that we have two processes P1 and P2. The process P1 wants to execute its critical section. P1 gets into the entry section. Since the value of lock is 0 hence P1 changes its value from 0 to 1 and enters into the critical section.

1. Meanwhile, P1 is preempted by the CPU and P2 gets scheduled. Now there is no other process in the critical section and the value of lock variable is 0. P2 also wants to execute its critical section. It enters into the critical section by setting the lock variable to 1.
2. Now, CPU changes P1's state from waiting to running. P1 is yet to finish its critical section. P1 has already checked the value of lock variable and remembers that its value was 0 when it previously checked it. Hence, it also enters into the critical section without checking the updated value of lock variable.
3. Now, we got two processes in the critical section. According to the condition of mutual exclusion, morethan one process in the critical section must not be present at the same time. Hence, the lock variable mechanism doesn't guarantee the mutual exclusion.
4. The problem with the lock variable mechanism is that, at the same time, more than one process can see the vacant tag and more than one process can enter in the critical section. Hence, the lock variable doesn't provide the mutual exclusion that's why it cannot be used in general.
5. Since, this method is failed at the basic step; hence, there is no need to talk about the other conditions to be fulfilled.