# SNS COLLEGE OF TECHNOLOGY

**Coimbatore-35.**
**An Autonomous Institution**

**Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade**
**Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai**

## COURSE NAME : OPERATING SYSTEMS

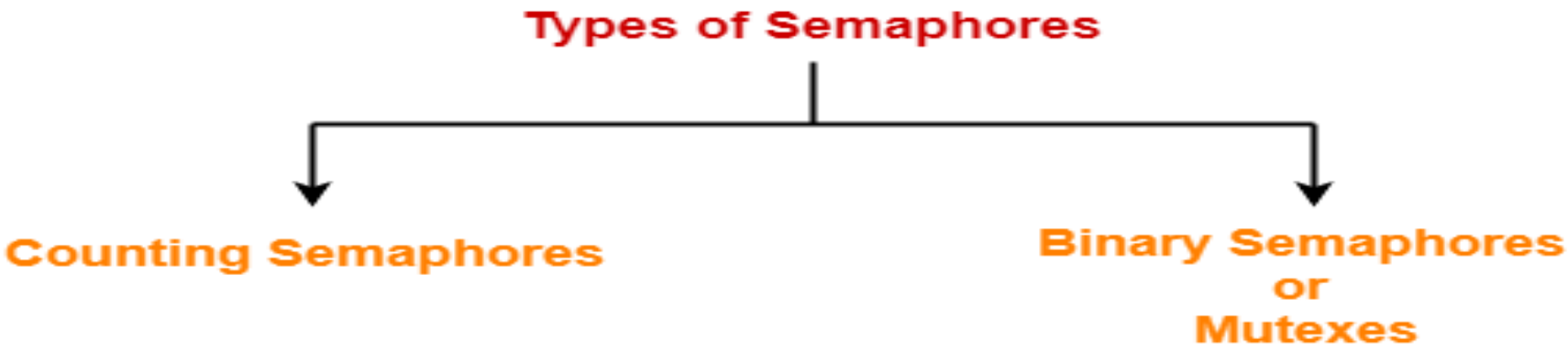
## II YEAR/ IV SEMESTER

## UNIT – II PROCESS SCHEDULING AND SYNCHRONIZATION

## Topic: Synchronization hardware – Semaphores

Dr.B.Vinodhini
Associate Professor
Department of Computer Science and Engineering

# *Semaphores*




- The main disadvantage of the semaphore – ***Busy Waiting***
- While a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code
- Busy waiting wastes CPU cycles that some other process might be to use productively
- This type of semaphore is also called a ***spinlock*** because the process spins while waiting for the lock

# Semaphores

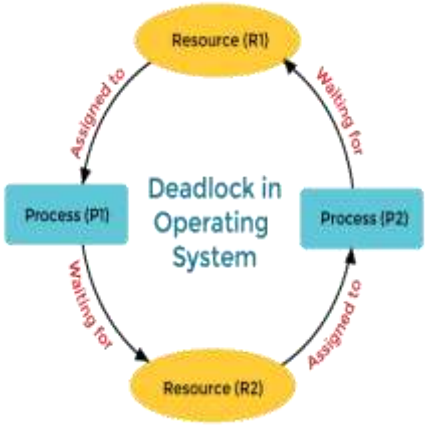
To overcome the need for busy waiting, we can modify the definition of the wait () and signal () semaphore operations.

- When a process executes the wait () operation and finds that the semaphore value is not positive, it must wait.
- However, rather than engaging in busy waiting, the process can block itself.
- The block operation places a process into a waiting queue associated with the semaphore, and the state of the process is switched to the waiting state.
- Then control is transferred to the CPU scheduler, which selects another process to execute.

# Disadvantages of Semaphores-Dead lock and Starvation

- The implementation of a semaphore with a waiting queue may result in a situatio where two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes.
- The event in question is the execution of a signal() operation. When such a state i reached, these processes are said to be deadlocked.

| P0 | P1 |
|---|---|
| wait (S); | wait (Q); |
| wait (Q); | wait (S); |
| . | . |
| . | . |
| . | . |
| signal (S); | signal (Q); |
| signal (Q); | signal (S); |




- Deadlock and starvation are conditions in which the processes requesting a resource have been delayed for a long time.
- Deadlock happens when every process holds a resource and waits for another process to hold another resource.
- In contrast, in starvation, the processes with high priorities continuously consume resources, preventing low priority processes from acquiring resources.

# Hardware Based Solution-Test and Set Lock

Atomic Operation

All are happened in Single operation without any Interrupt

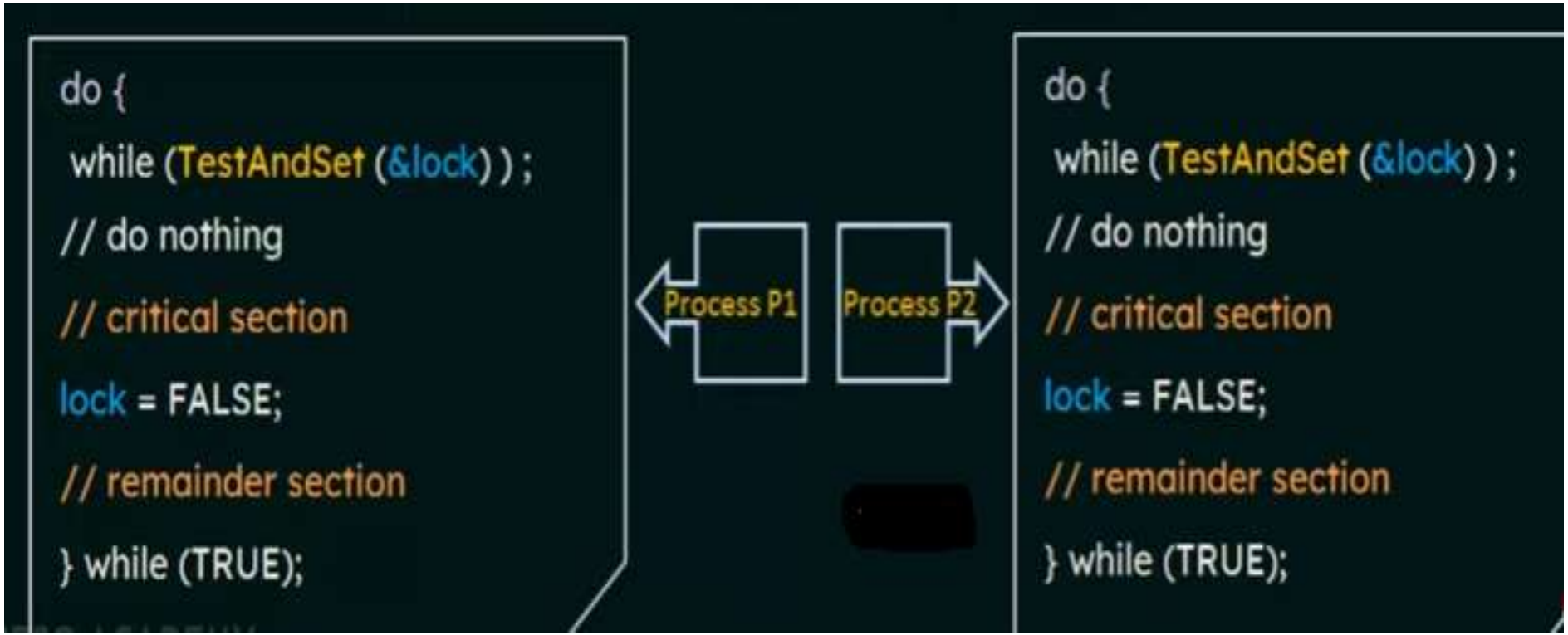
- A hardware solution to the critical section Problem
- There is a shared variable which can take either of the two values 0 or 1
- Before Entering into the critical section a process inquires about the lock
- If it is locked it keeps on waiting till it becomes free
- If it is not locked it takes the lock and executes the critical section

```
boolean TestAndSet (boolean *target) {
boolean rv = *target;
*target = TRUE;
return rv;
}
```

The definition of the TestAndSet () instruction

Lock value set to 0-If it is Unlocked
Lock Value set to 1-It is locked

# Hardware Based Solution-Test and Set Lock




***Satisfies Mutual Exclusion***

***Does Not Satisfy Bounded Waiting***

# *References*


1. Silberschatz, Galvin, and Gagne, “Operating System Concepts”, Ninth Edition, Wiley India Pvt Ltd, 2009.
2 . Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition, Pearson Education, 2010.


Thank
you