CS & IT ENGINEERING

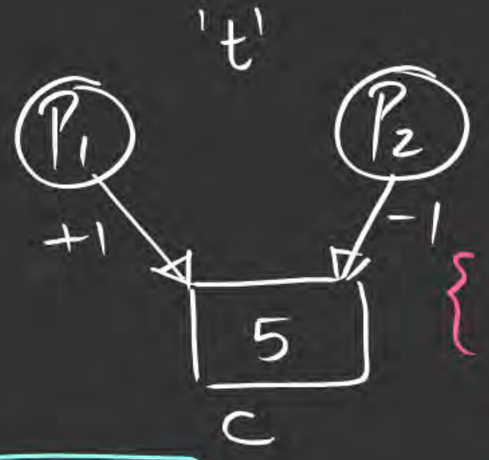
Operating System

Process Synchronization

Lecture No. 3

By- Dr. Khaleel Khan sir

- Critical Section Problem
- Requirements of CS Problem
- Lock Variable




```
{                          {
    C = C+1;                   C = C-1;
}                          }
```

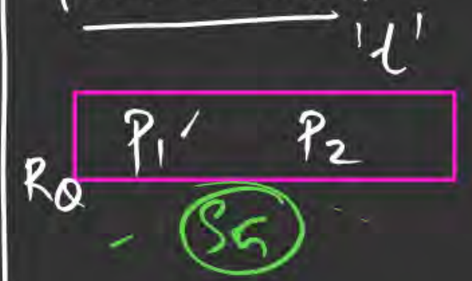
I. Load R_1, C;
II. Inc R_1;
III. Store C, R_1

I. Load R_2, C;
II. Dcr R_2;
III. Store C, R_2;

(Any Process that executes in user Mode, Can get pre Empted after any Instruction)

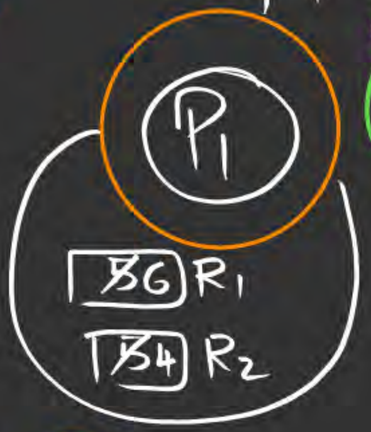
UNIV. ASSUMPTION

Scenario:



C 5 6

CPU



Inconsistency

't' : P1 : I; II; P1

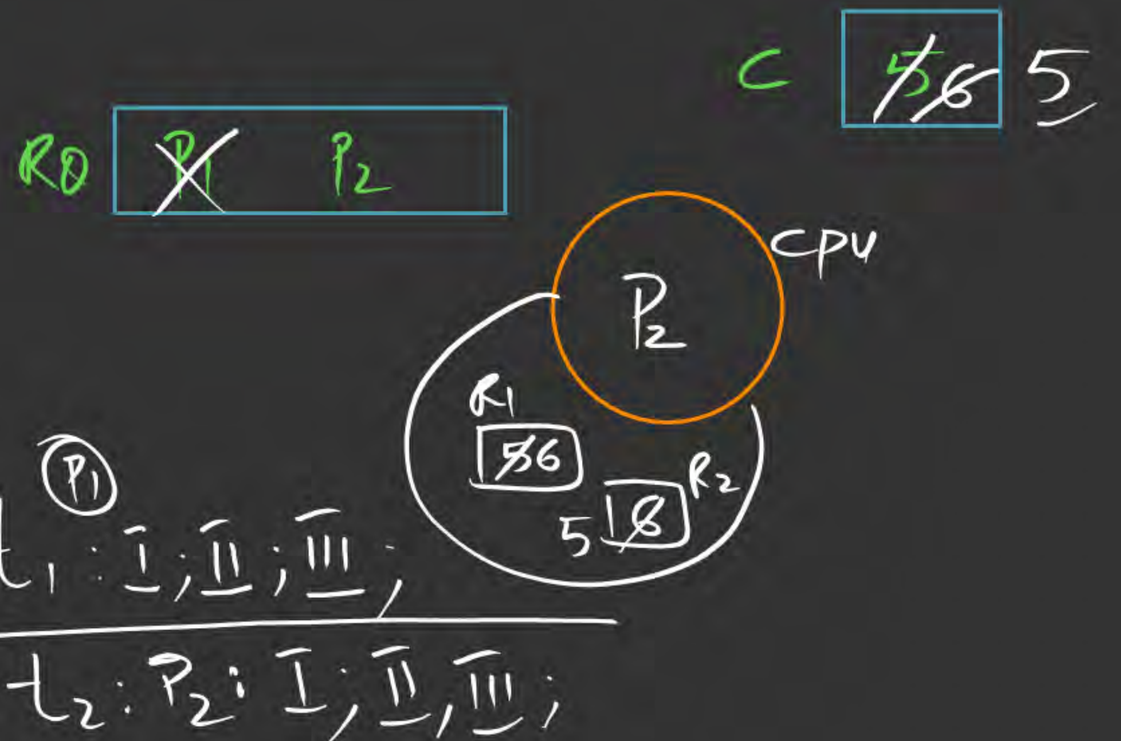
't1' : P2 : I; II, III

't2' : P1 : III

When?

(Some Times we get Correct result & other Times, it is possible to get Incorrect results)

As a End user we want a Solution that gives always correct result <whether Preemption takes place or not>



Impl. of Producer-Consumer Problem

```
#define N 100
int Count = 0;   // No. of Data items in Buffer
int Buffer[N];
```

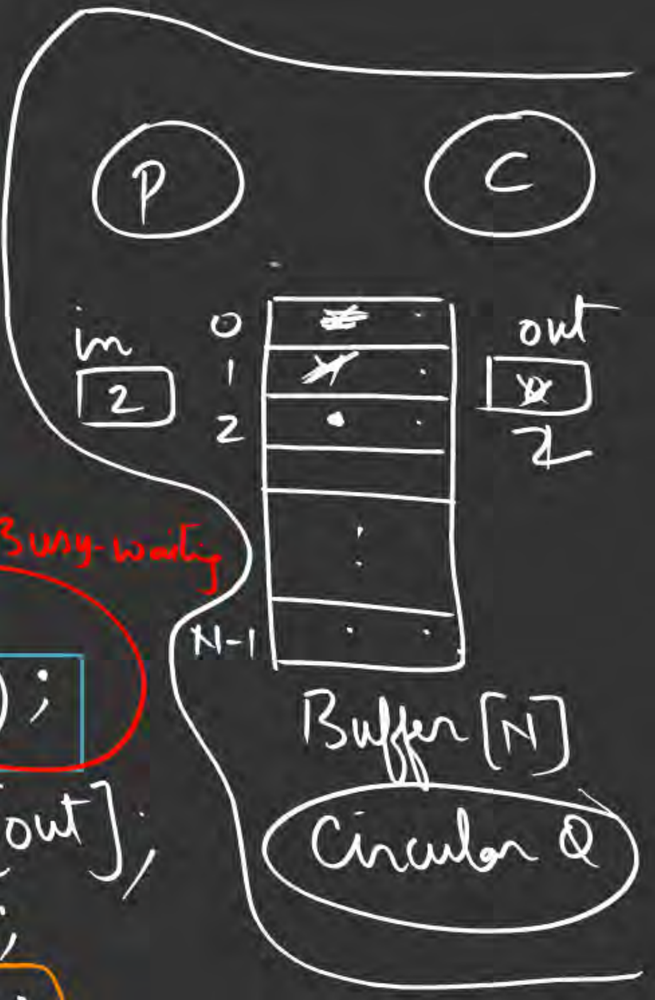
Both

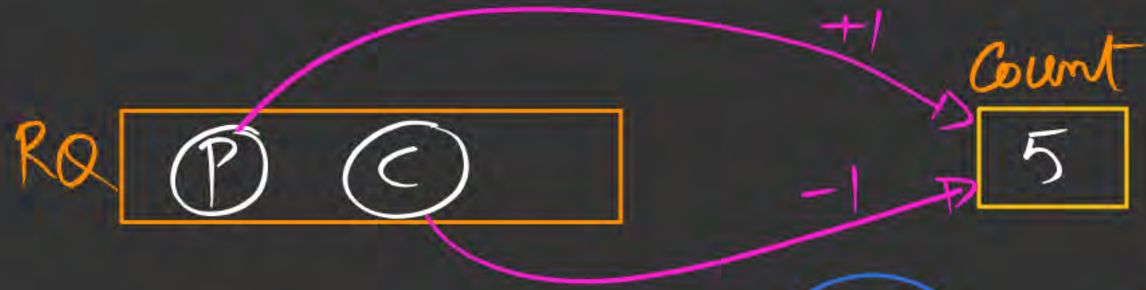
Busy waiting

```
void Producer(void)
{
    int itemp, in = 0;
    while (1)
    {
NCS     a) itemp = Produce_item();
CS      b) while (Count == N);
CS      c) Buffer[in] = itemp;
CS      d) in = (in + 1) % N;
CS      e) Count = Count + 1;
    }
}
```

```
void Consumer(void)
{
    int itemc, out = 0
    while (1)
    {
        a) while (Count == 0);
        b) itemc = Buffer[out];
        c) out = (out + 1) % N;
        d) Count = Count - 1;
        e) Process_item(itemc);
    }
}
```

Busy-waiting





N = 100

: Competition

CPU

Inconsistency

't'

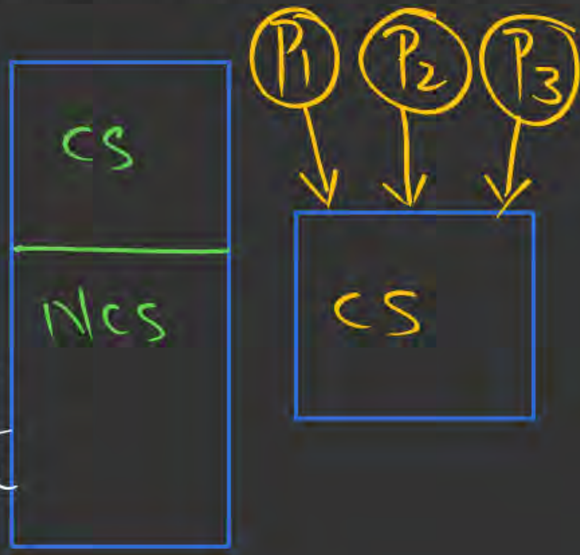
Necessary conditions for Synch Problems

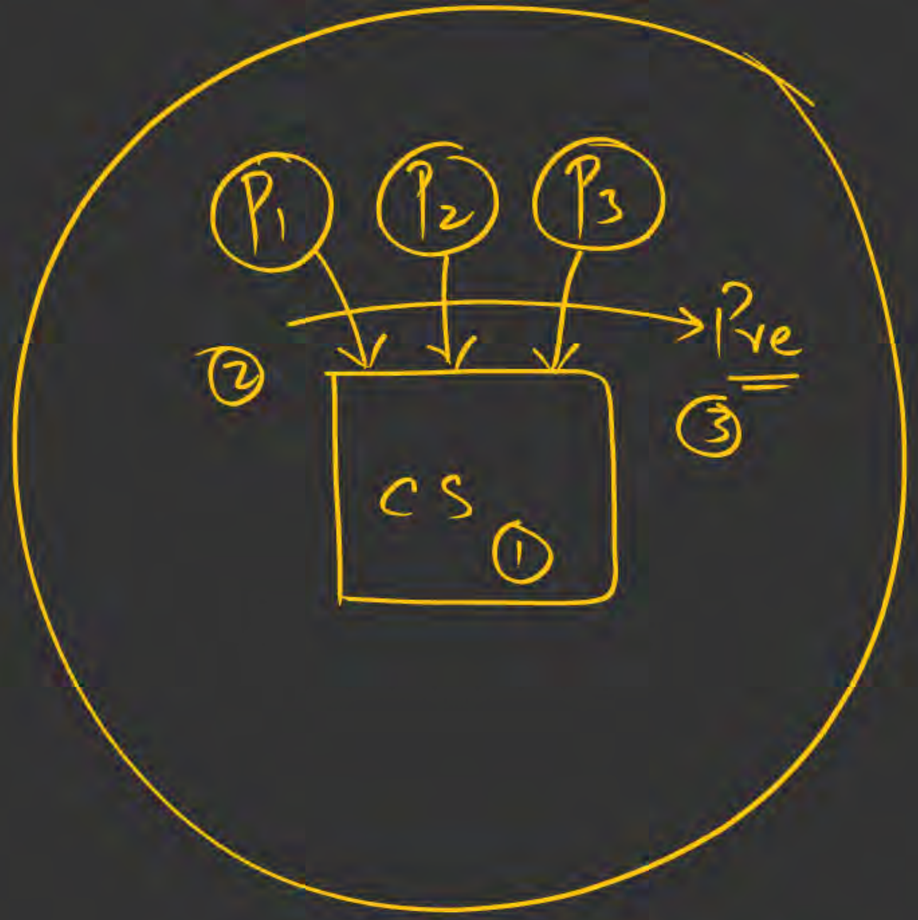
1) **Critical Section (CS)** : is that Part of the Program where Shared resources are accessed;

Non-critical Section (NCS) / **other Section** / **Remainder Sec** : Part of the Program that does not access Shared resource;

2) **Race condition** : Situation wherein processes are trying to access <CS> & final result depends on the order in which they finish their update

3) **PreEmption** :

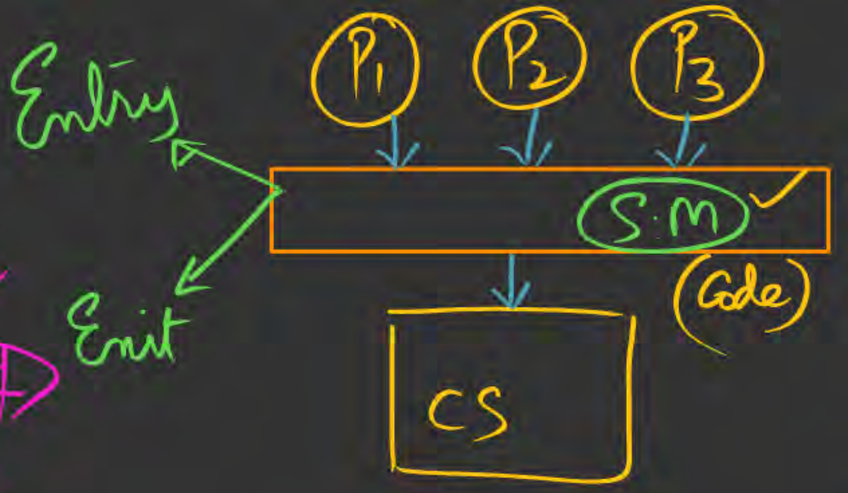




Critical Section Problem

Soln

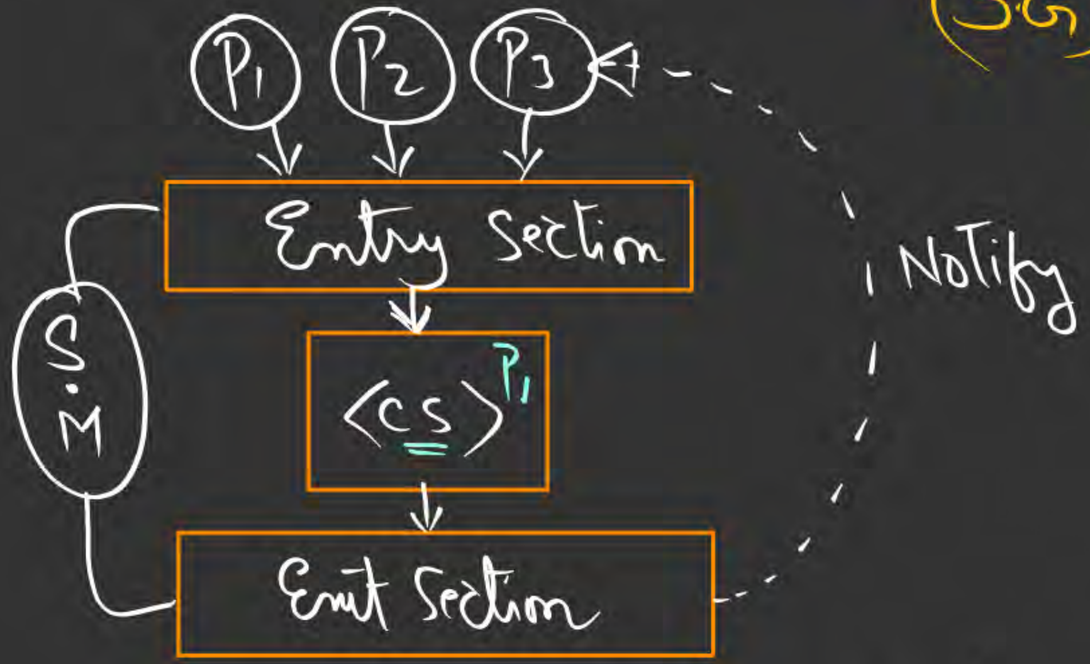
Algo



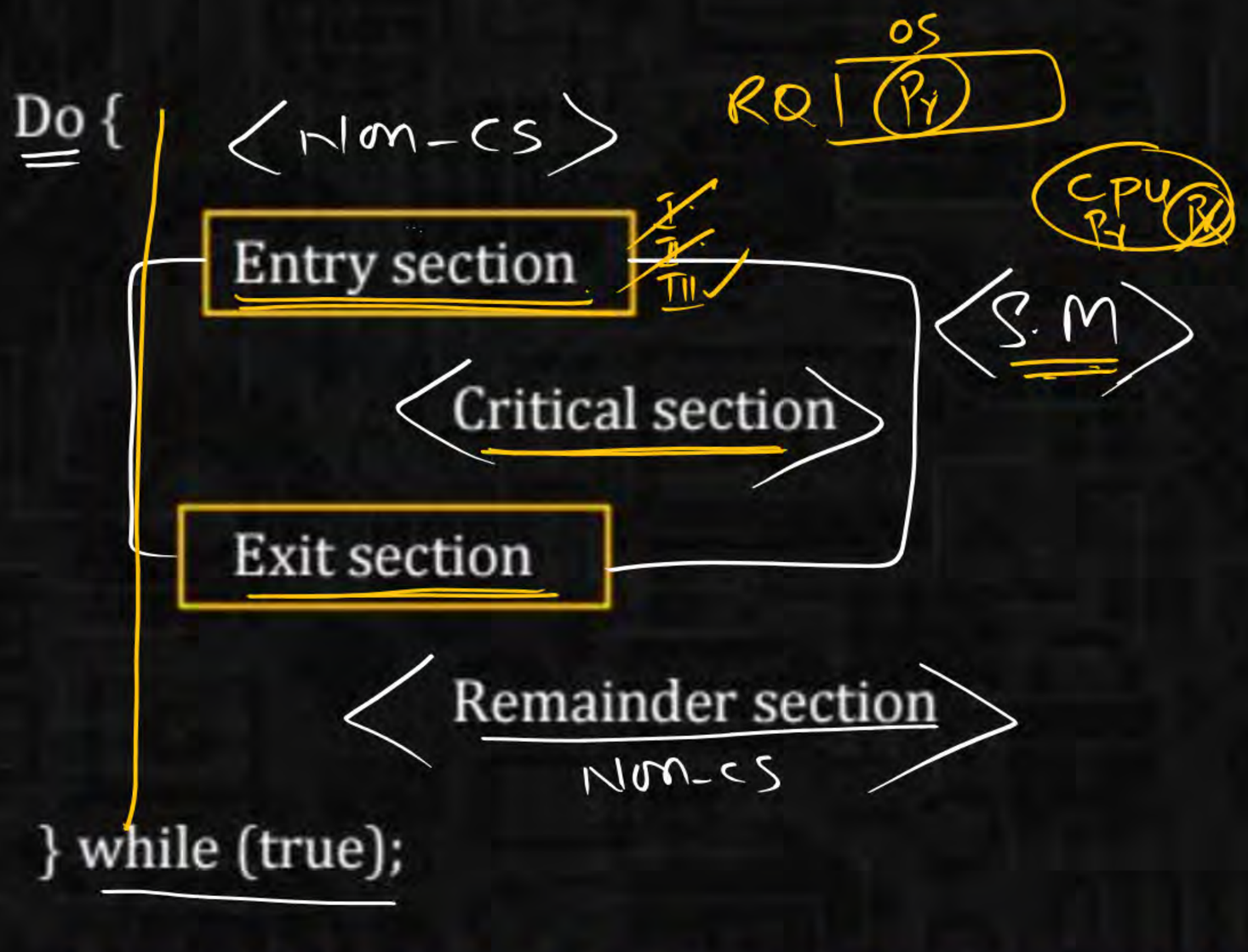
Synch Tool

Synch Mech. (SM)

(SG)



<Architecture/model of S.M>



Process running in User Mode can get PreEmpted any-time & any Number of times <Anywhere (after completing any Instn)>

General Structure of a Typical Process with Synch. Mechanism

Requirements of C.S Problem (to be satisfied by Synch. Mechanism)

(i) Mutual Exclusion (M/E): (SM) ✓

No Two Processes Should be Present in <CS> together at the same time;

4-Queens($q_1 .. q_4$)

<4-Queens>

(ii) Progress:

(iii) Bounded waiting:

1. A solution to the Critical-Section Problem must satisfy the following three requirements: Mutual Exclusion: If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.

Critical section ✗

Critical section ✗

Critical section ✓

2. Progress: If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.

3. Bounded waiting: There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.


THANK YOU
GW
SOLDIERS !