Peterson's Solution

Two process *solution*

Assume that the LOAD and *STORE instructions* are atomic; *that is, cannot be interrupted.*

The two processes *share two variables:*

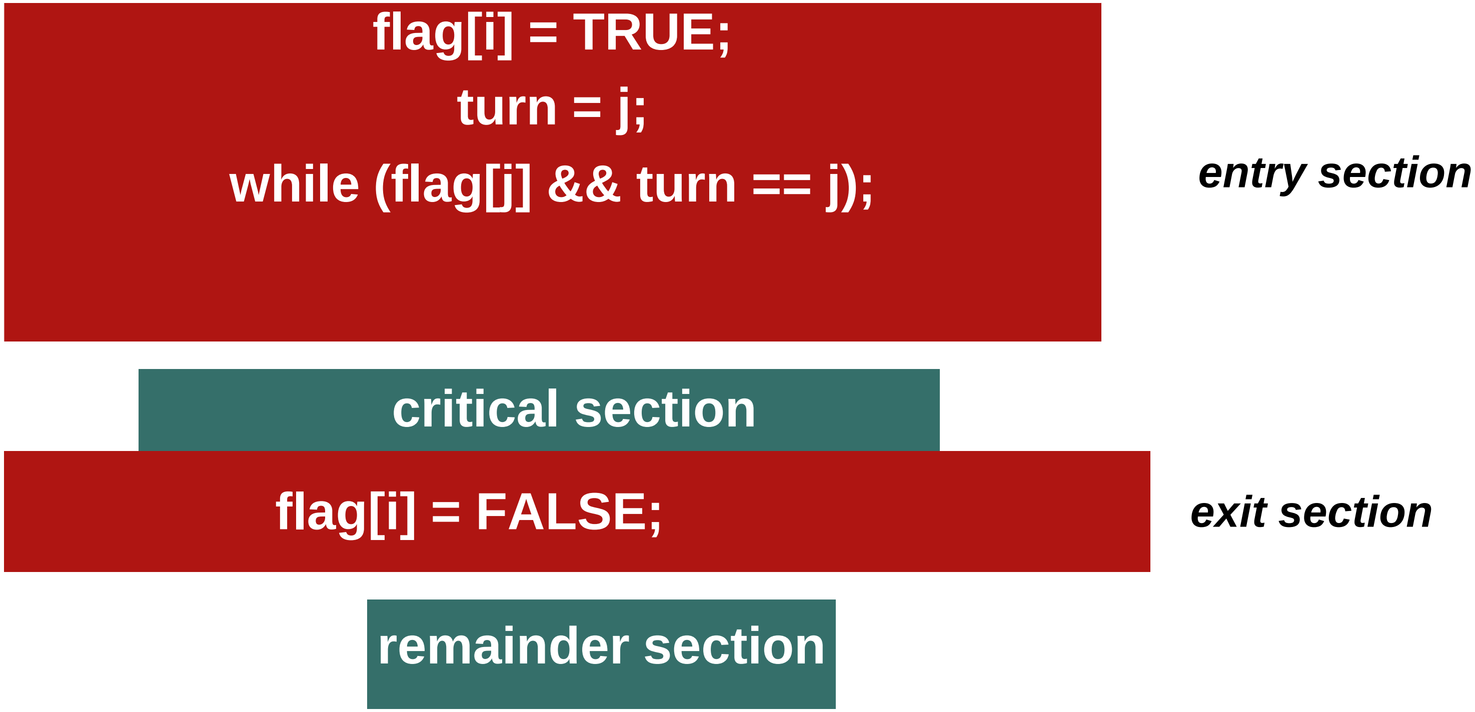
int **turn;**

Boolean **flag[2]**

The variable **turn** *indicates whose turn it is to enter the critical section.*

The **flag** *array is* used to *indicate if* a process *is* ready *to enter the critical section.* **flag[i] =** *true implies that* process P_I *is* ready!

Algorithm for Process P_i



```
do {
        flag[i] = TRUE;
        turn = j;
        while (flag[j] && turn == j);
            critical section
        flag[i] = FALSE;
            remainder section
} while (1)
```

entry section

exit section

Two processes *executing concurrently*

PROCESS 1

```
do {
        flag1 = TRUE;
        turn = 2;
        while (flag2 && turn == 2);
        critical section.....
        flag1 = FALSE;
        remainder section.....
} while (1)
```

PROCESS 2

```
do {
        flag2 = TRUE;
        turn = 1;
        while (flag1 && turn == 1);
        critical section .....
        flag2 = FALSE;
        remainder section.....
} while (1)
```

Shared Variables

flag1, flag2
turn

EXAMPLE

Process 0:

flag[0] := TRUE
turn := 1
check (flag[1] = TRUE and turn = 1)
- Condition is false because flag[1] = FALSE
- Since condition is false, no waiting in while loop
- Enters the critical section

Phase-1

Process 1:

flag[1] := TRUE
turn := 0
check (flag[0] = TRUE and turn = 0)
- Since condition is true, it keeps busy waiting until it loses the processor
- Process 0 resumes and continues until it finishes in the critical section

Phase-2

Process 0:

- Leaves critical section
  Sets flag[0] := FALSE
- Start executing the remainder (anything else a process does besides using the critical section)
- Process 0 happens to lose the processor

Phase-3

Process 1:

check (flag[0] = TRUE and turn = 0)
- This condition fails because flag[0] = FALSE - No more busy waiting - Enter the critical section

Phase-4