

process synchronization

- 2 or more processes (threads)
- share memory
- cooperate

$$x = 0$$

process 1

∞ loop

{ work

⋮

$x = x + 1$

print x

}

1 2 3 4

1 3 2

process 2

∞ loop

{ work

⋮

$x = x + 1$

print x

}

$x=0$



$x = x + 1$



$x = x + 1$

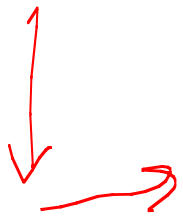


print x

atomic →

load $x \rightarrow r_{reg}$

$inc(r_{reg} + 1)$



store $r_{reg} \rightarrow x$

2 cpus → maybe '1'

1 cpu

Critical section problem

CS

C.S. \rightarrow $\left[\begin{array}{l} \text{enter CS} \\ n = n + 1 \\ \text{exit CS} \end{array} \right.$

- ① mutual exclusion
- ② progress
- ③ bounded waiting

mutual exclusion

- not more than 1 process will be executing inside a critical section at any point in time
- if no process is in CS and a process α wants to enter CS then it should be allowed to enter CS
- if a process wants to enter then there has to be a limit on the # of times another process is allowed to enter - - -

- Can we build critical sections using only software & no atomic instructions?

- NO

- maybe

- yes ; Dekkers solution

- yes & its really simple

- Peterson's solution

15
or 20
yrs



~~flag[0] flag[1] → set to false~~

~~process 0~~

flag — 0 = unlocked
— 1 = locked

process:

while (flag == 1);

flag = 1

critical section

flag = 0

(A)

(B)

Same

flag[0] & flag[1] ← 0

process 0

```
flag[0] = 1
while flag[1] == 1 {
  { CS }
}
flag[0] = 0
```

```
flag[0] = 0;
flag[0] = 1;
}
```

flag[0] flag[1]

process 0

flag[0] = 1

turn = 1

while (flag[1] == 1
and turn == 1);

CS

flag[0] = 0

process 1

flag[1] = 1

turn = 0

while
(flag[0] == 1
and
turn == 0);

CS

flag[1] = 0;