CSE 330: Operating Systems

Class: 10 **Date:** 9/20

Fall 2016

Note Title

- Critical section problem
- Software
- Hardware solutions.

Uniprocessor Hardware

- Disable interrupts

ATOMIC

- Enable interrupts

Does not work on multiproc

- missed interrupts
- crashes
- CS is global
- Need privileged mode

Multiprocessor hardware solution

- one atomic instruction

  - Test & set (Intel has XCHG)

→ Test&set variable ← in memory

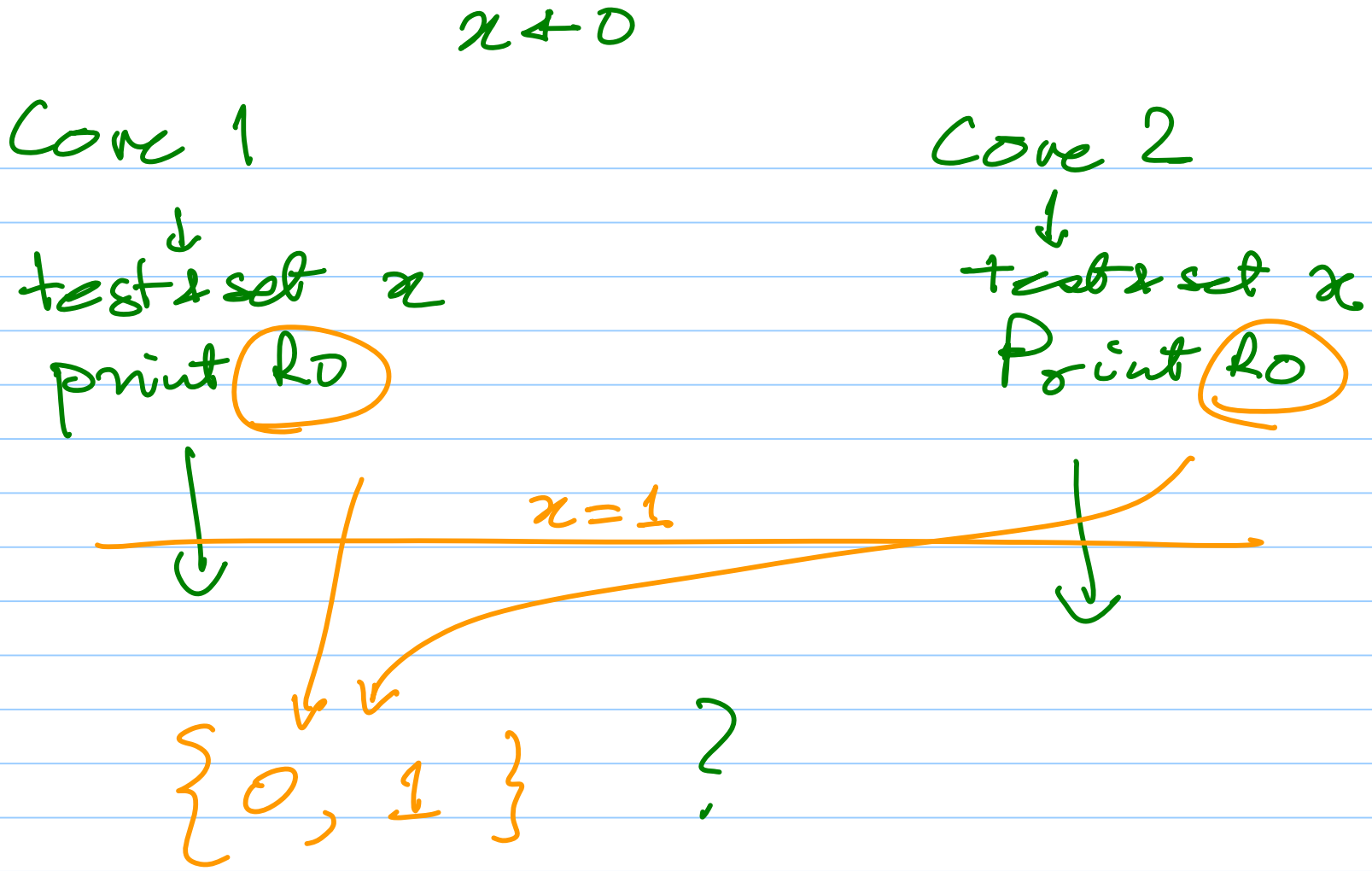
one instruction (atomic)

↓

Test&set x

```
R0 ← x
x ← 1
```

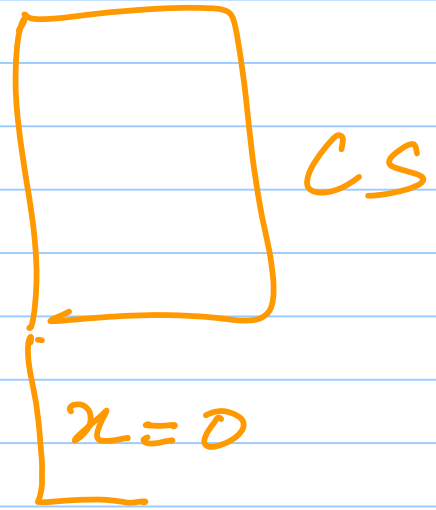
atomic


x ← 0
Core 1
test&set x
print R0
Core 2
test&set x
Print R0
x = 1
{0, 1} ?

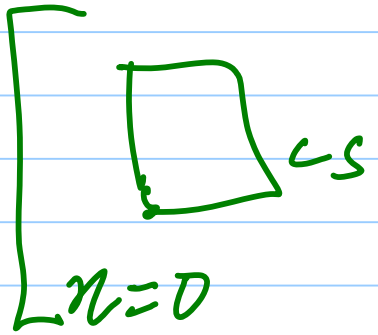
R0 = 1

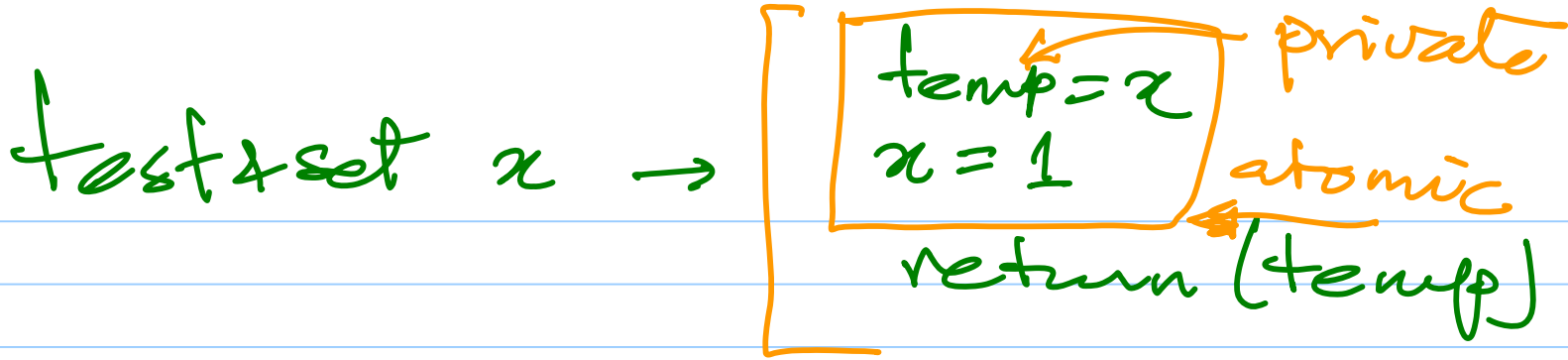
x = 0

while (R0 == 1) {test x set x }



while (R0 == 1) test set x





while test&set (x) ;

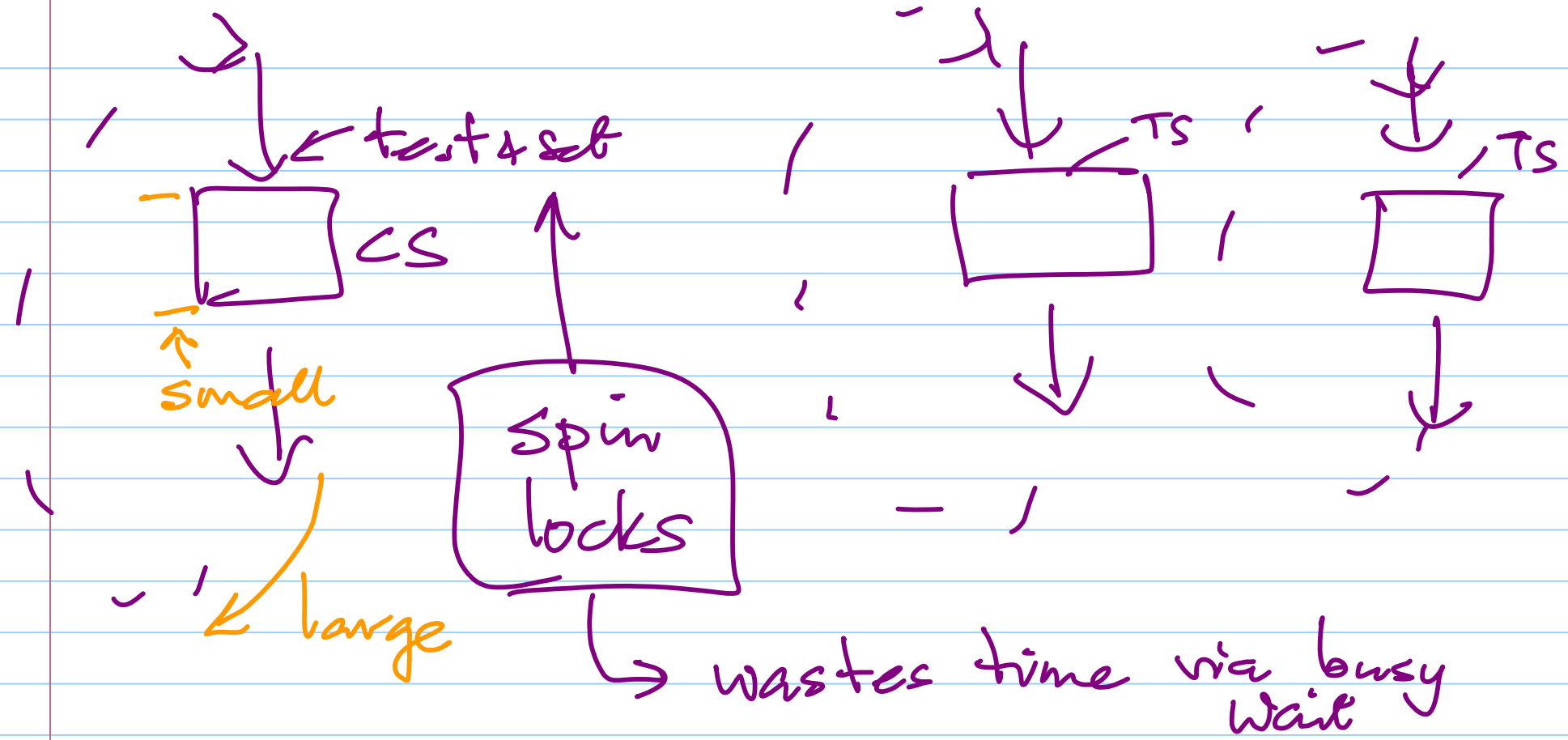
Critical section

x = 0

✓ – mutex

✓ – progress

X – bounded wait


test & set
CS
small
large
spin locks
TS
TS
wastes time via busy wait

- Using test+set (spin lock)
  on uniprocessors
  - ↳ it works
  - ↳ very wasteful
  - ↳ not advised.

→ Semaphores [Dijkstra, 1962 or 63]

- value (integer)
- data type

Sem S1, S2, S3;

init S1, S2, S3 to 1

| init
| P (wait)
| V (signal)

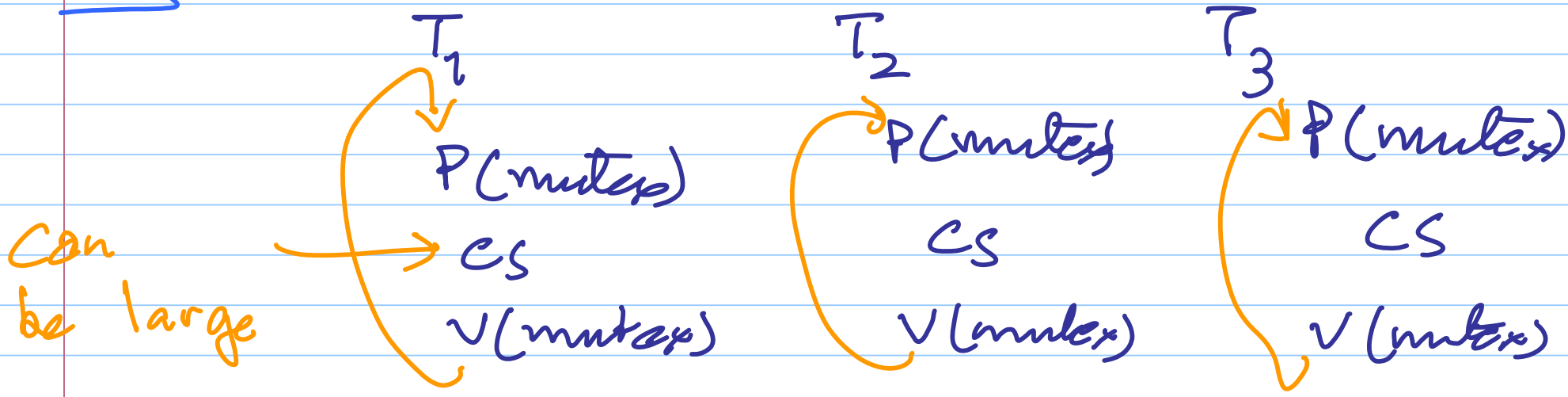
P(S1) → { while (S1 == 0) ; S1 --; } ← Atomic

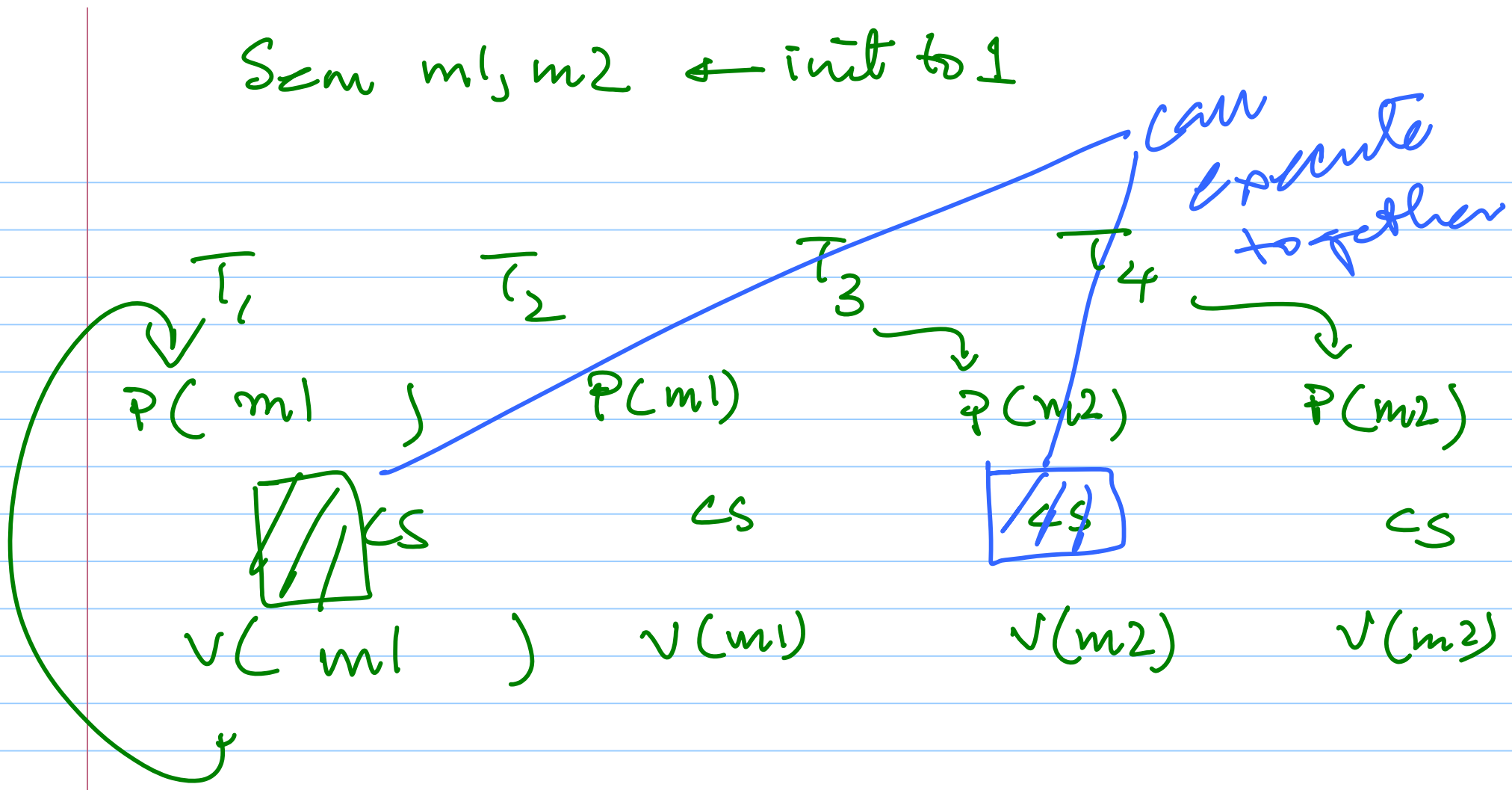
V(S1) → { S1 ++ } ← Atomic

Semaphore usage — mutex

declare → Sem mutex;

init → initSem(mutex, 1)




Sem m1, m2 ← init to 1
can execute together
T1
T2
T3
T4
P(m1)
P(m1)
P(m2)
P(m2)
CS
CS
CS
CS
V(m1)
V(m1)
V(m2)
V(m2)

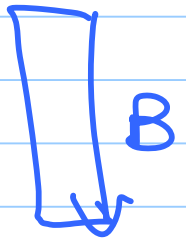
T_1

$P(m_1)$

$V(m_1)$

T_2

$P(m_2)$



$V(m_2)$

T_3

$P(m_1)$

$P(m_2)$

$V(m_2)$

$V(m_2)$

T_1

① P(m1)

P(m2)

T_2

② P(m2)

P(m1)

if

1 & 2 occur → deadlock will occur

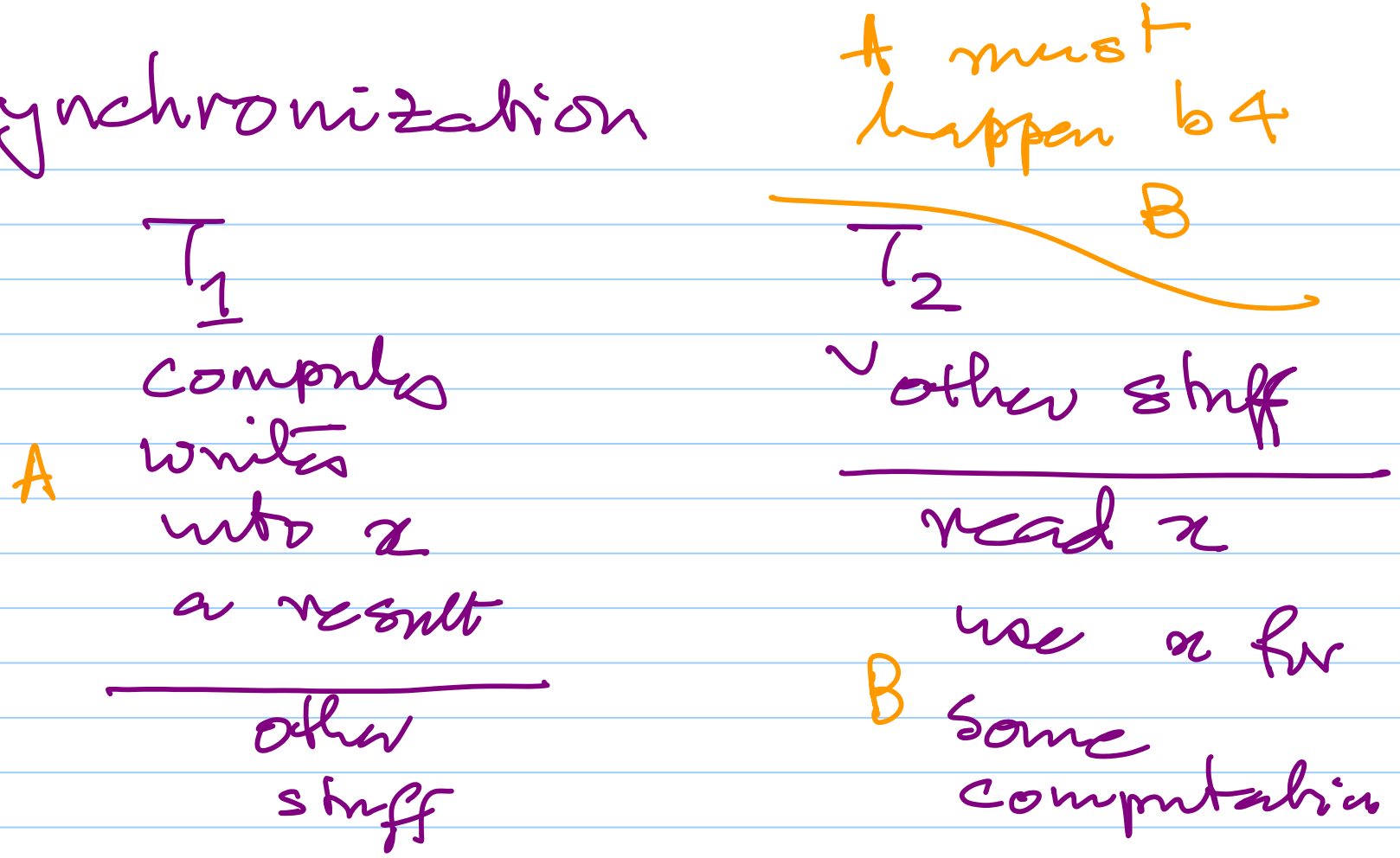
V(m2)

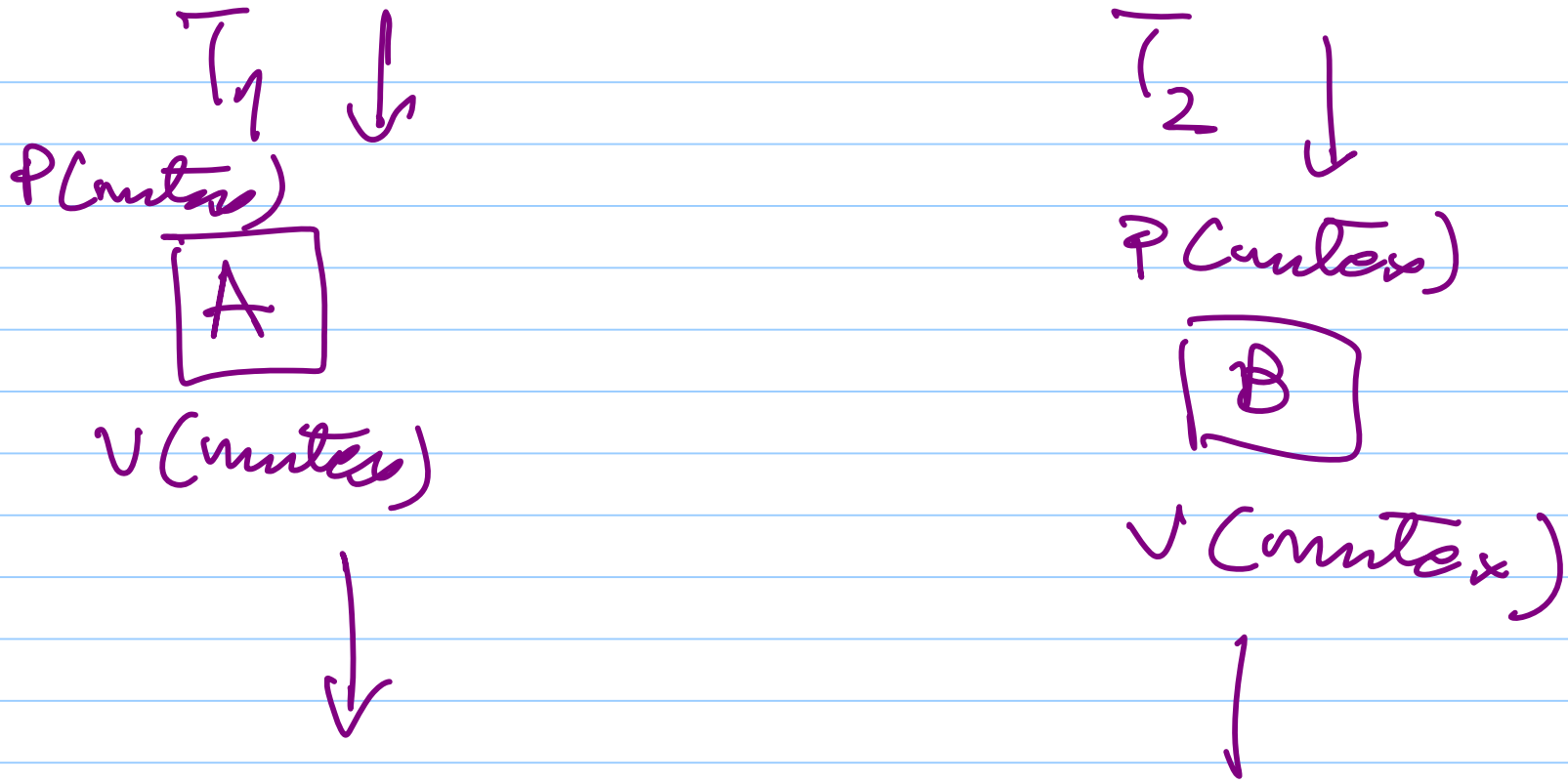
V(m1)

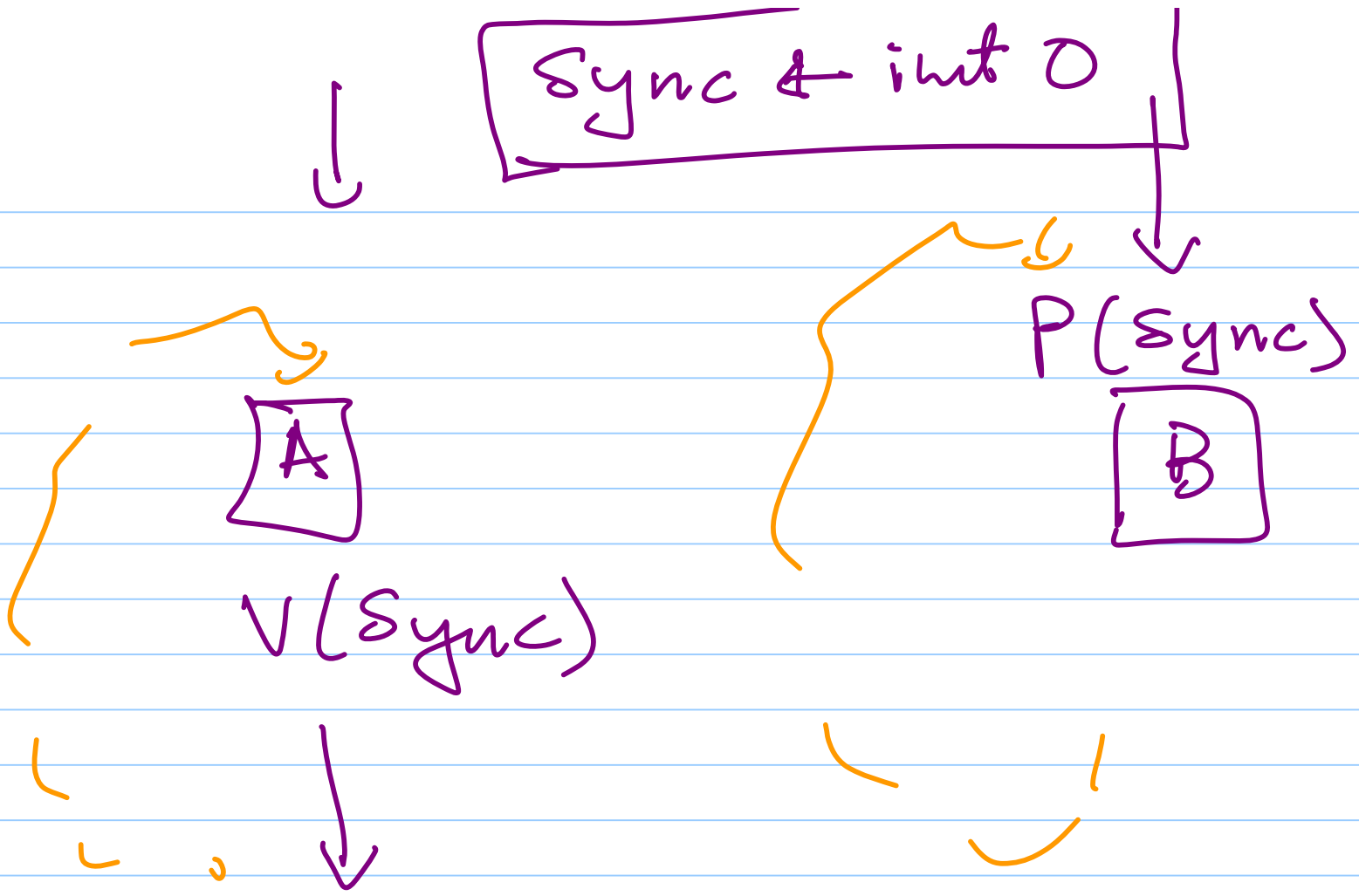
V(m1)

V(m2)

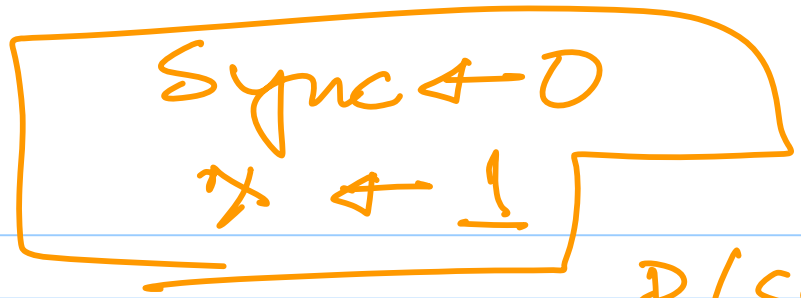
Synchronization




T1
P(mutex)
A
V(mutex)
T2
P(mutex)
B
V(mutex)


Sync ← init 0
P(Sync)
A
B
V(Sync)

Alternate



P(x)

A

V(sync)

P(sync)

B

V(x)