CSE 330: Operating Systems

Class: 11 **Date:** 2/13

Spring 2018

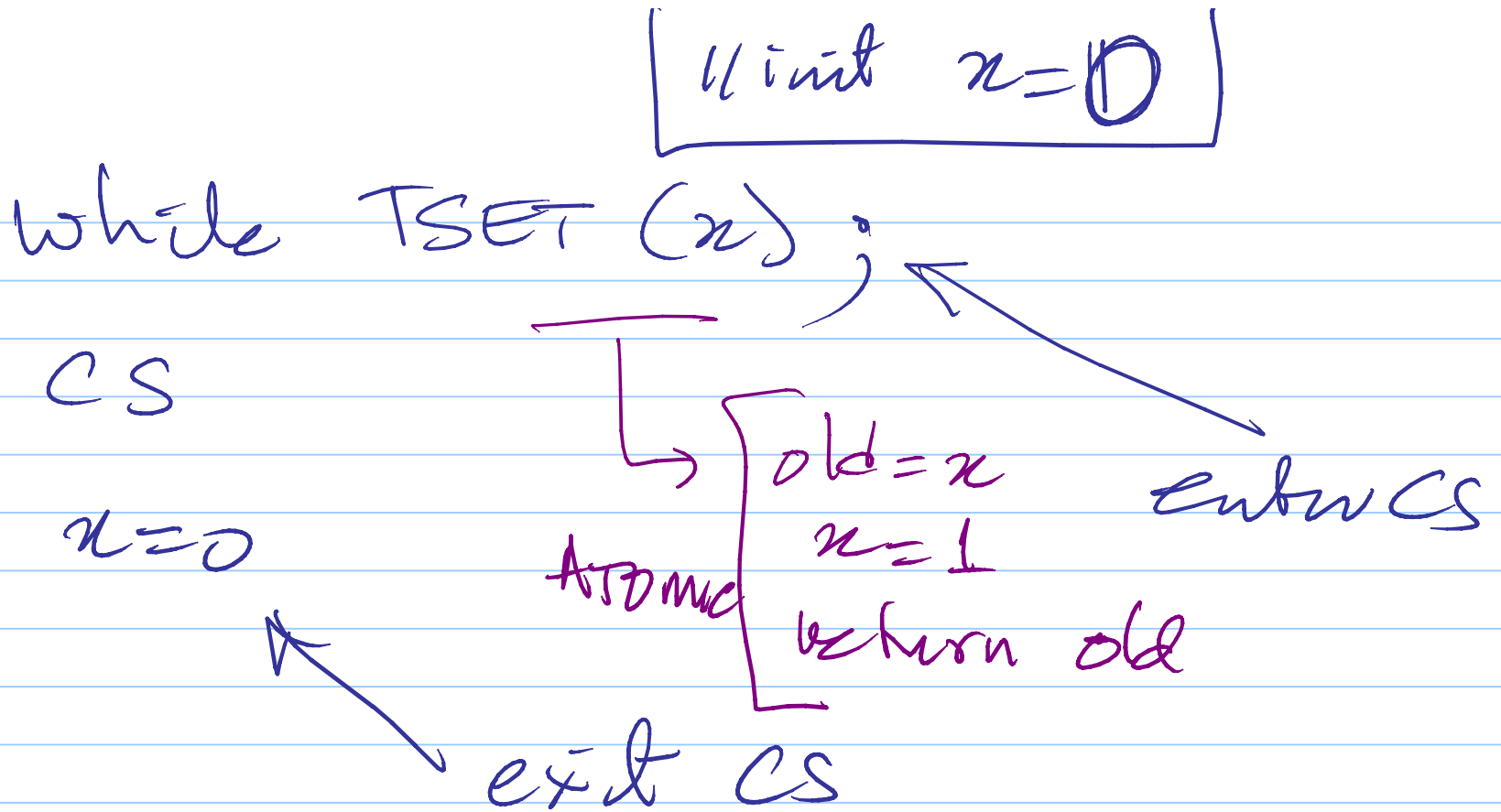
Note Title

- Critical Section — 1, 2, 3

- Software solutions

- actual (hardware solutions)

  ↳ Disable INT / Test & Set


init x=0
while TSET(x);
CS
x=0
old=x
x=1
return old
ATOMIC
enter CS
exit CS

$x = 0$

exactly the same time

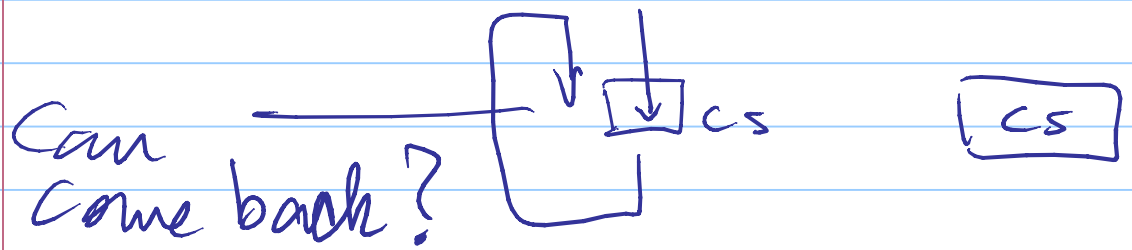
$y_1 = TSET(x)$ ⟷ $y_2 = TSET(x)$

---

x ~~=~~ 1

y_1, y_2 } at ~~least~~ most 1 zero

- Mutual Exclusion ✓
- Progress — only those who want to enter . . . ✓
- Bounded Waiting



while TSET(x);

CS

x = 0

while TSET(x);

CS

x = 0

- Test & set does not provide Bounded Waiting

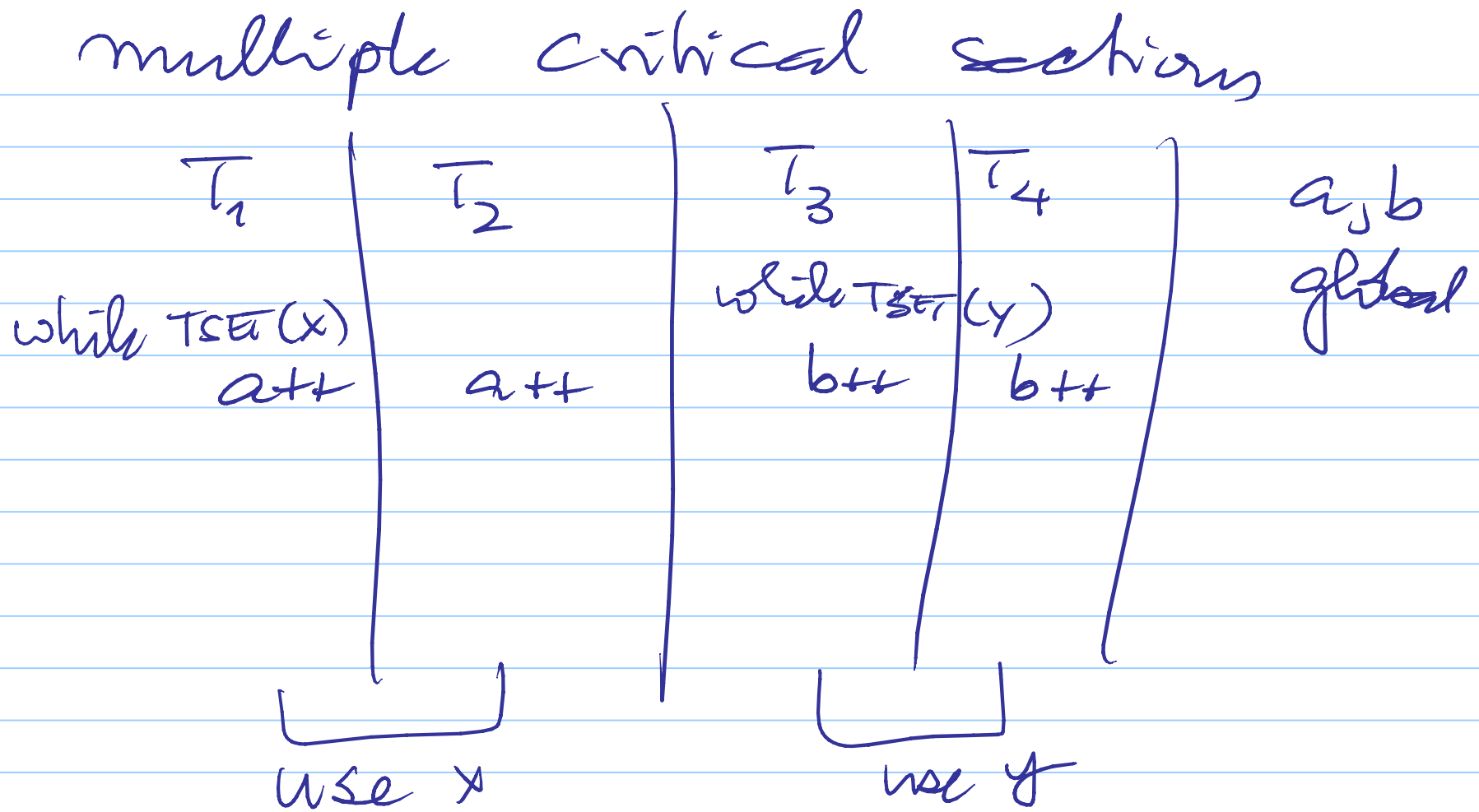
  BUT

  ① Keep CS very short

  ② most of the time CS is not busy

- Chance of starvation is small
- if overtaking happens — it resolves itself soon
* - also large CS is constructed out of small CS

multiple critical sections



Semaphores (another one is monitors)

↳ data type

↳ instances ...

methods or operations

↳ is an integer

methods

S is an instance of type semaphore

init (s, n) → s = n;

ATOMIC

(wait) P(s) → if [(s > 0) s--]

else goto

(signal) V(s) → [s++] ← ATOMIC

Use of Semaphores

① critical sections

Semaphore mutex; init(mutex, 1)

P(mutex)

CS

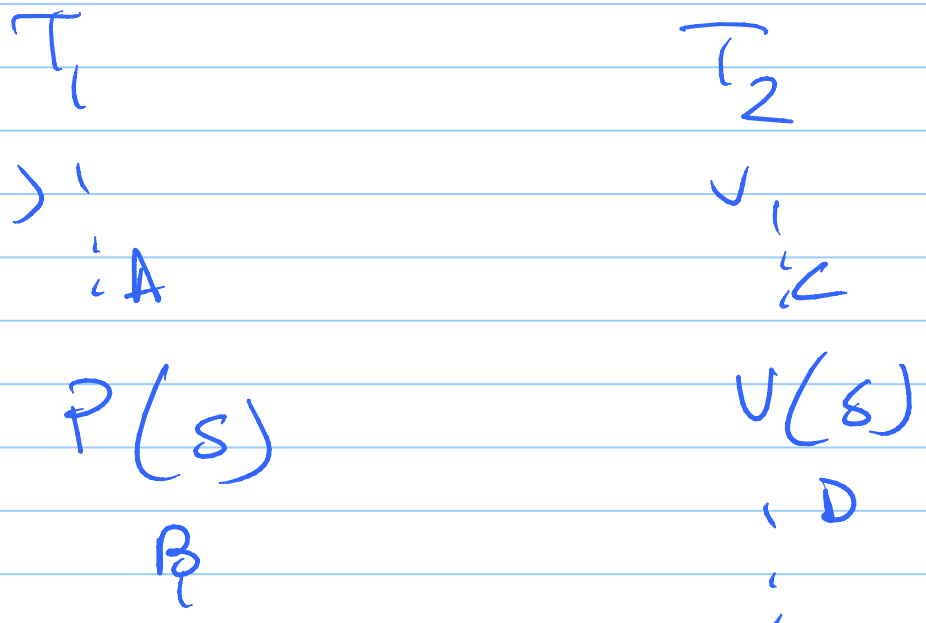
V(mutex)

Any number of processes / threads

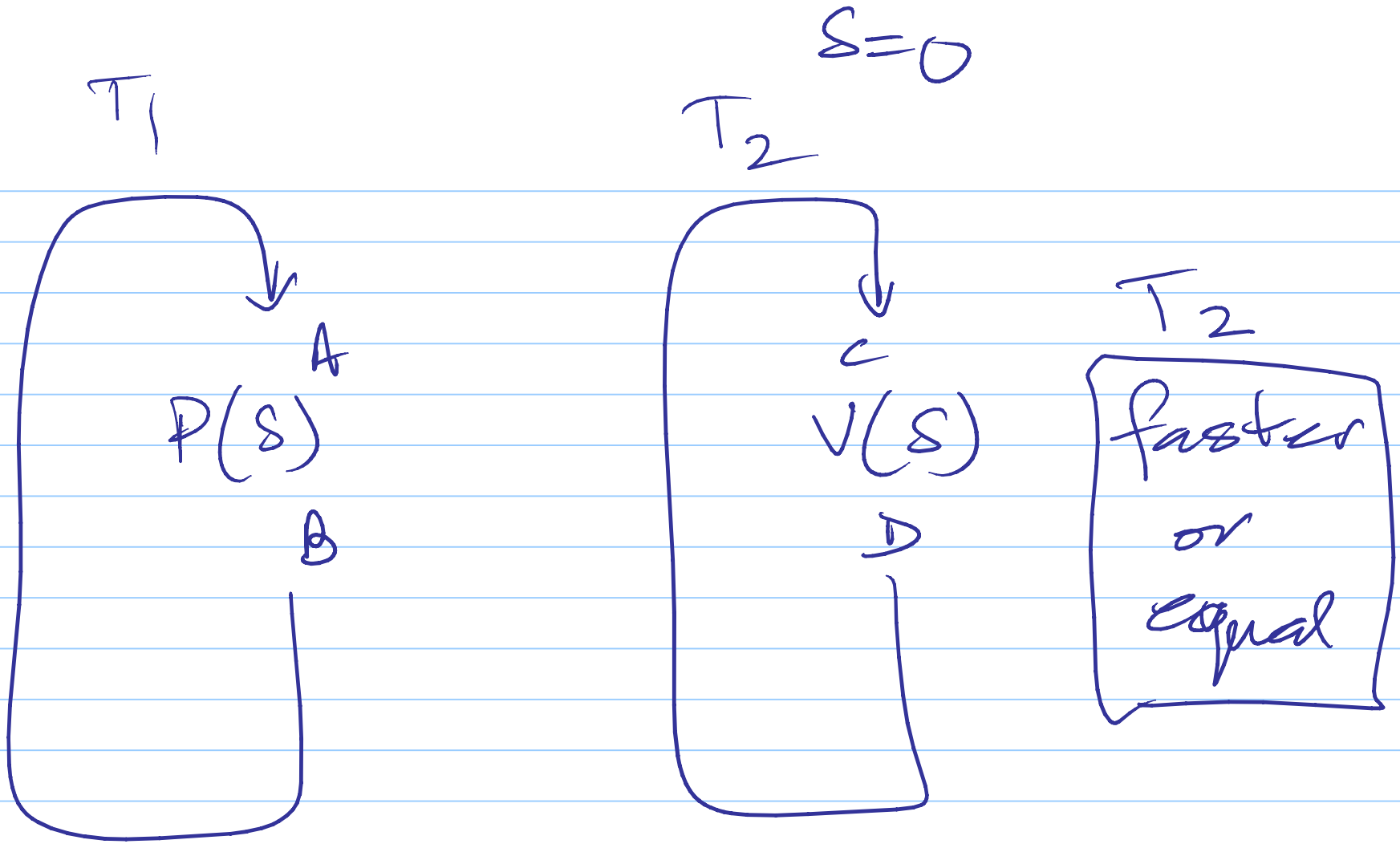
Semaphore S_1, S_2, S_3 ; ← init all to 1

P(S_1)	P(S_1)	P(S_2)	P(S_3)
CS	CS	CS	CS
V(S_1)	V(S_1	V(S_2)	P(S_3)

Synchronization [Sem S; S=0]



C will execute before B

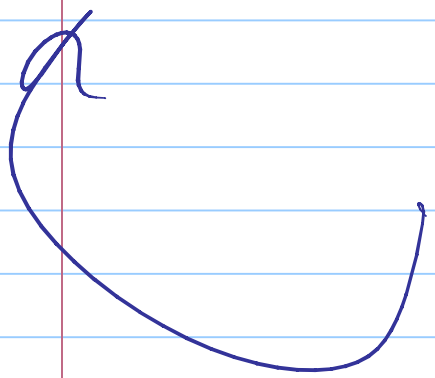

S=0
T1
T2
A
P(S)
B
C
V(S)
D
T2
faster
or
equal

$S_1 = 0$

$S_2 = 0$

$V(S_2)$

$P(S_1)$



$V(S_1)$

$P(S_2)$