CSE 531: Distributed and Multiprocessor Operating Systems

Class: 05 Date: 9/9

Fall 2020

Note Title

Locking —

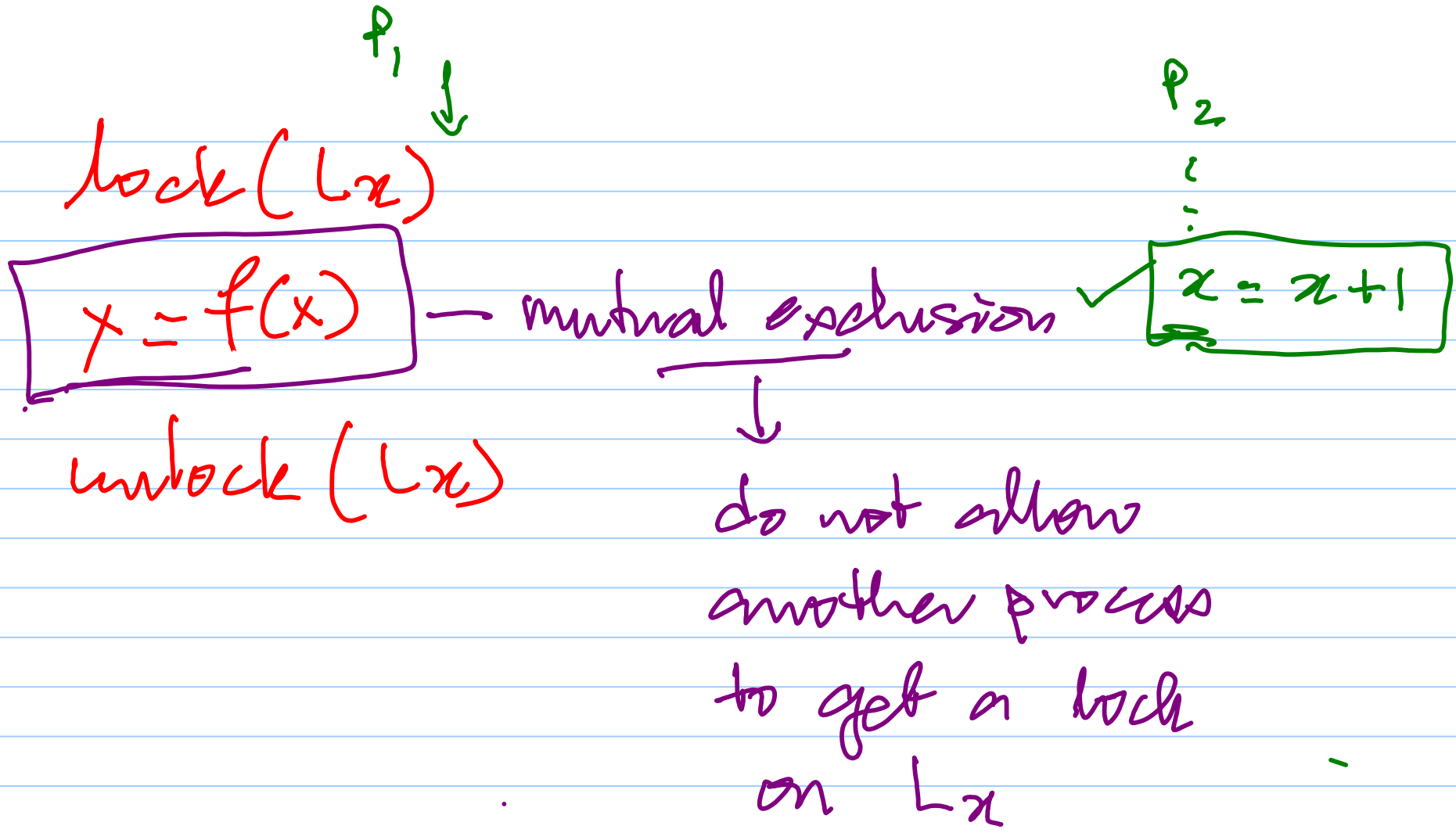
- advisory → low level

- mandatory

file system

P_1 → lock R

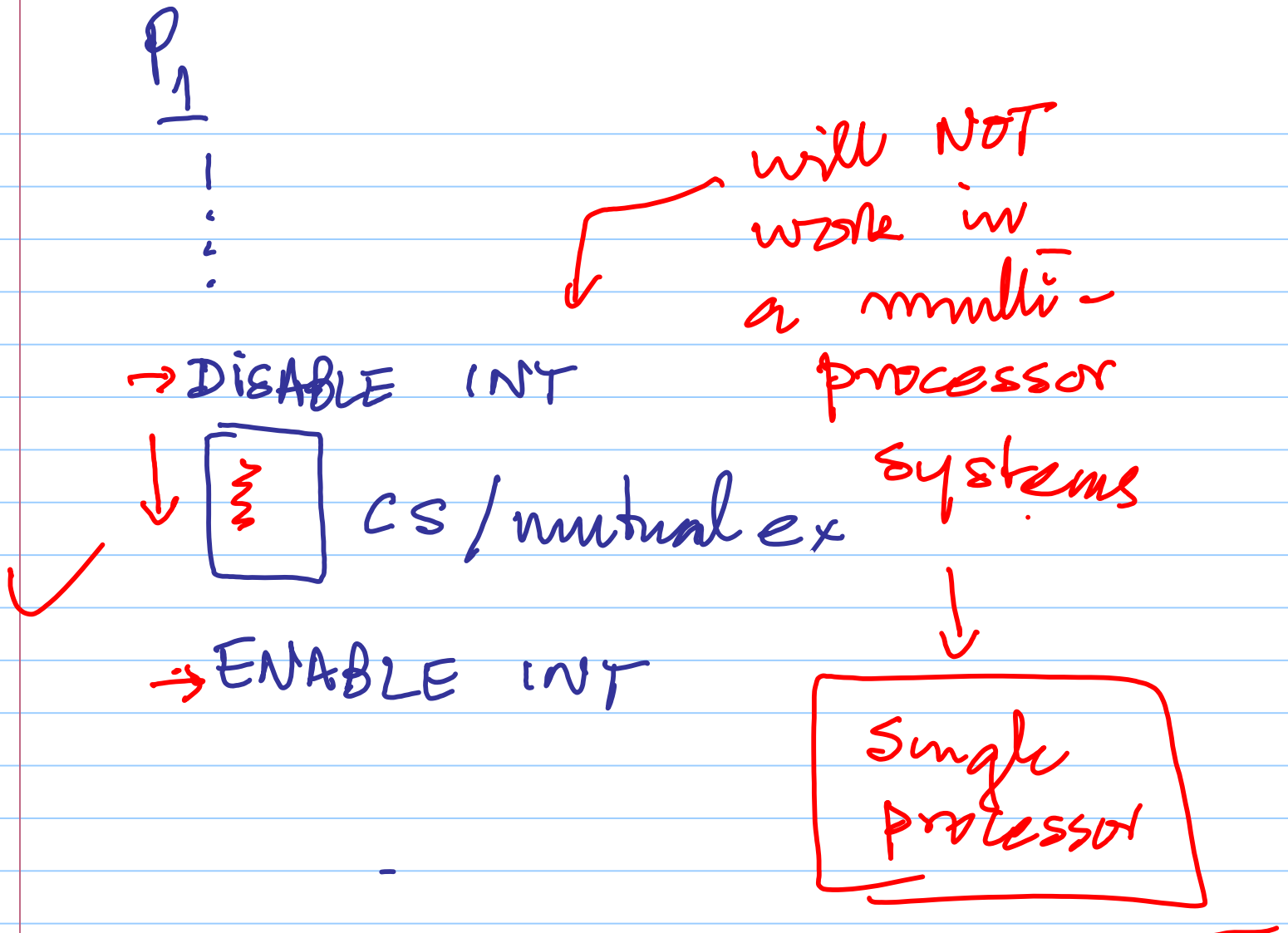
P_2 → Cannot access R.


P1
lock(Lx)
x = f(x)
unlock(Lx)
mutual exclusion
do not allow another process to get a lock on Lx
P2
x = x+1

Lock(var) →

Atomic instructions
- ① Test & set
- ② exchange ---

Disabling Interrupts


P1
DISABLE INT
CS / mutual ex
ENABLE INT
will NOT work in a multi-processor systems
Single processor

Critical section

= mutex → test & set ✓ - mutex

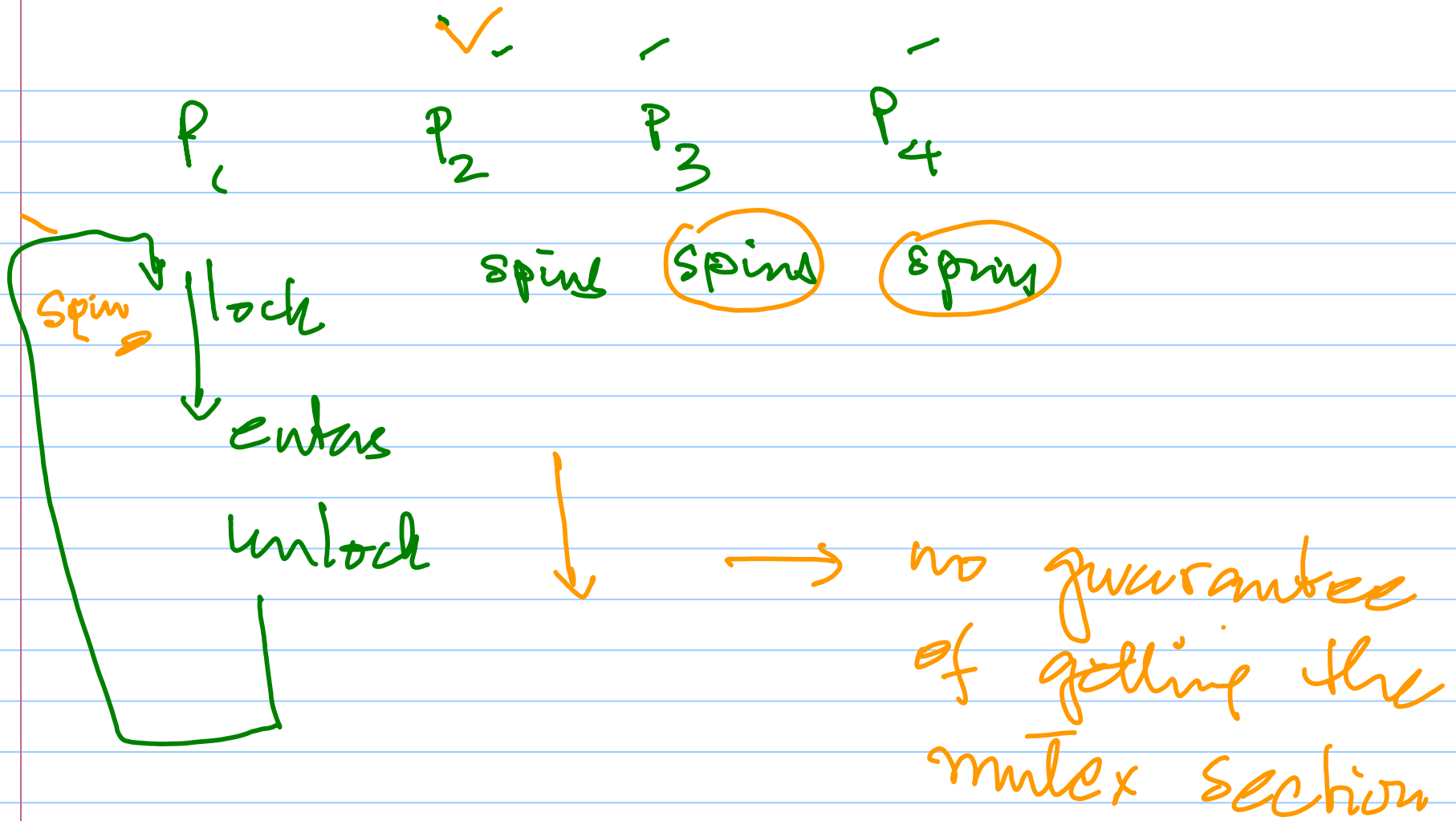
+

- progress ✓ - progress

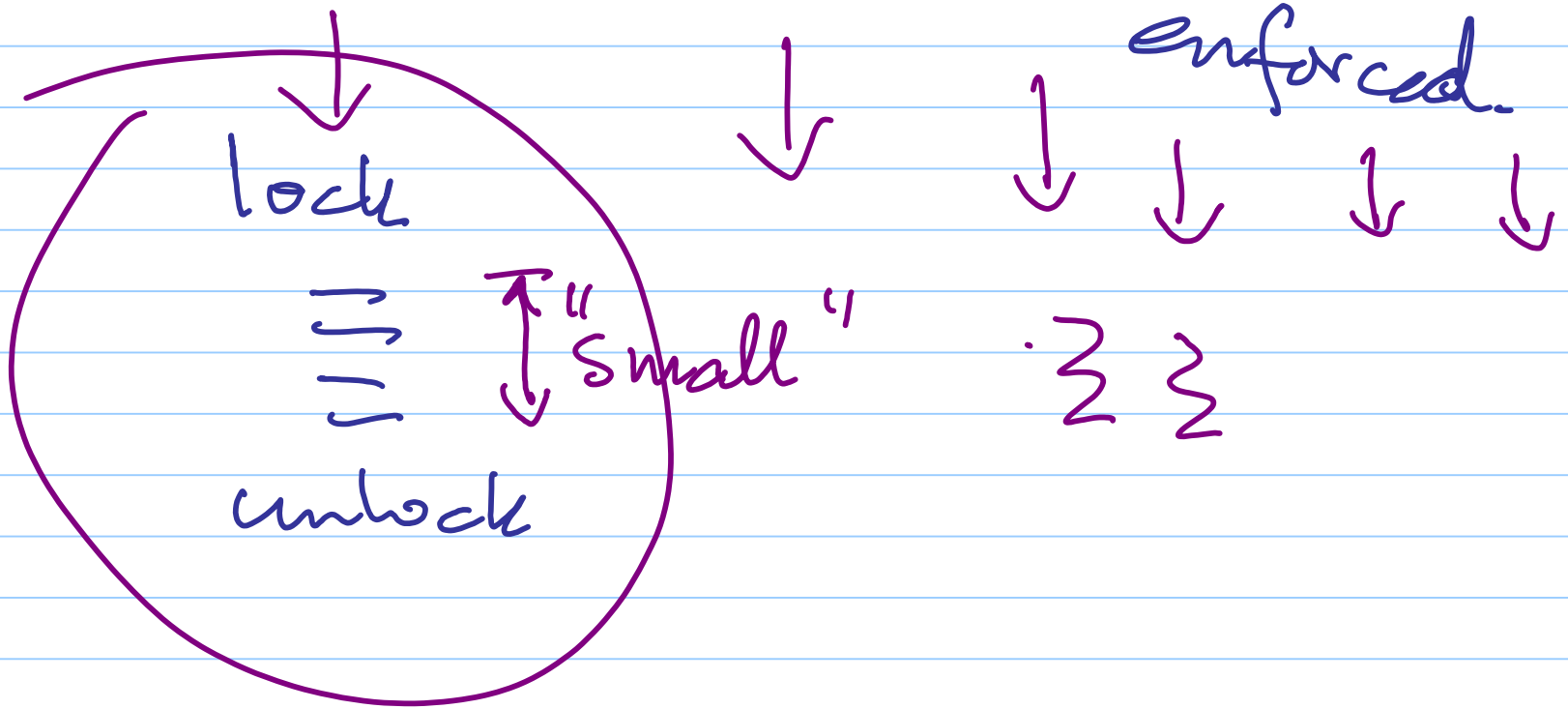
+

- Bounded waiting → fail

Lx=0
lock (lx)
unlock (lx)
P1
P2
Spin
lock (Lx)


P1
P2
P3
P4
spins
spins
spins
spin
lock
enters
unlock
no guarantee of getting the mutex section

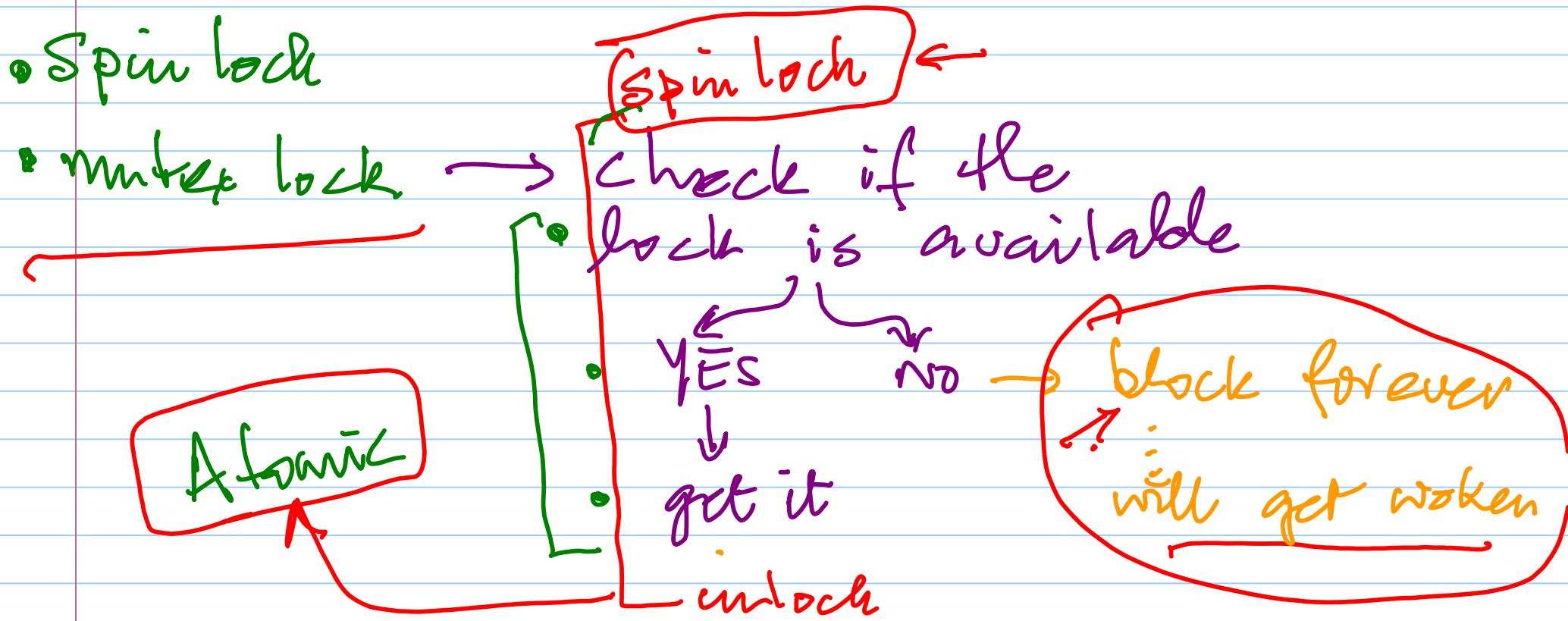
→ Spin lock sections should be small → programmer enforced.



Large critical sections

↳ do not use test & set

- Spin lock
- mutex lock



mutex lock on x

MLx { SLx → spin lock
Qx → queue
count_x → 0 = unlock
1 = locked

m_lock(MLx)

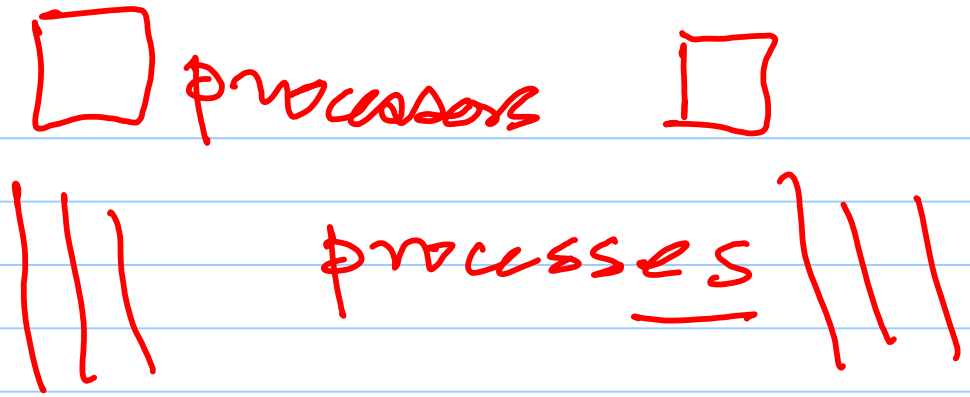
sp_lock(SLx)

if (count x == 0) count++

else block ×

sp_unlock(SLx)

for MLx



how do interrupts work?

+ processor modes

processor mode

- user
- privileged

(2)

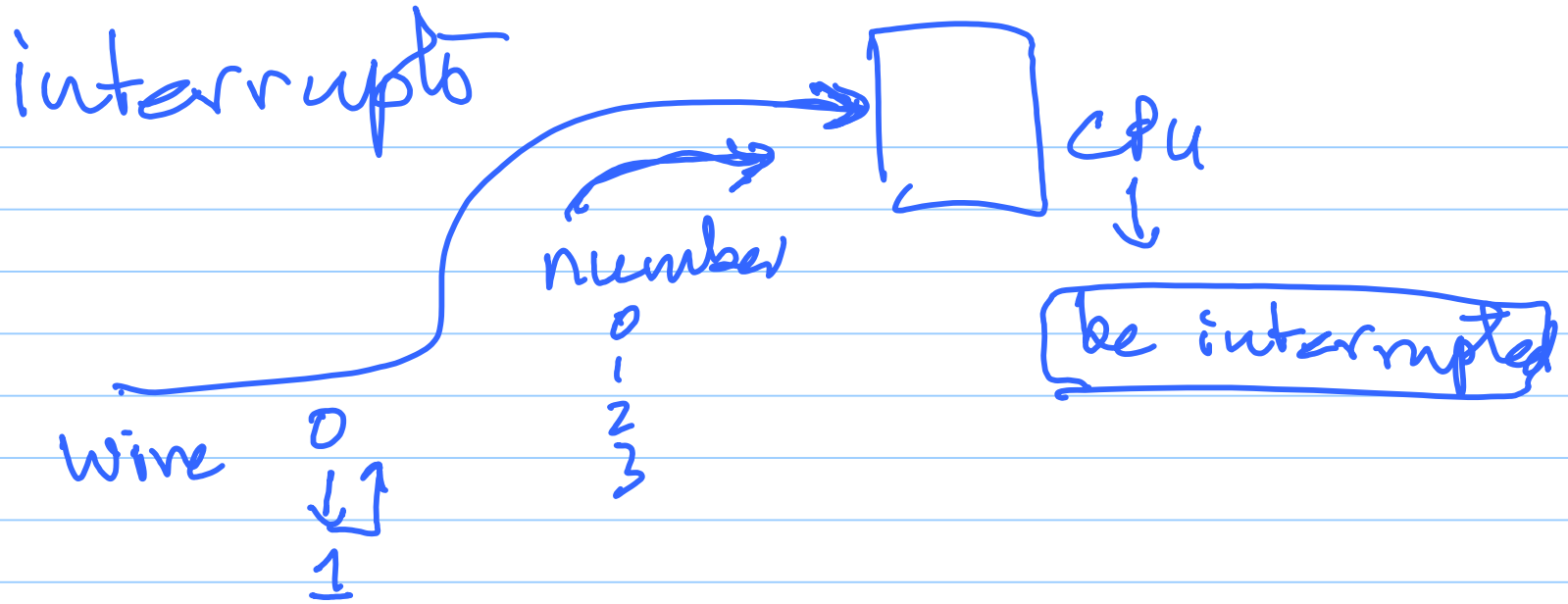
(4) rings

0
1
2
3

all

4 op codes

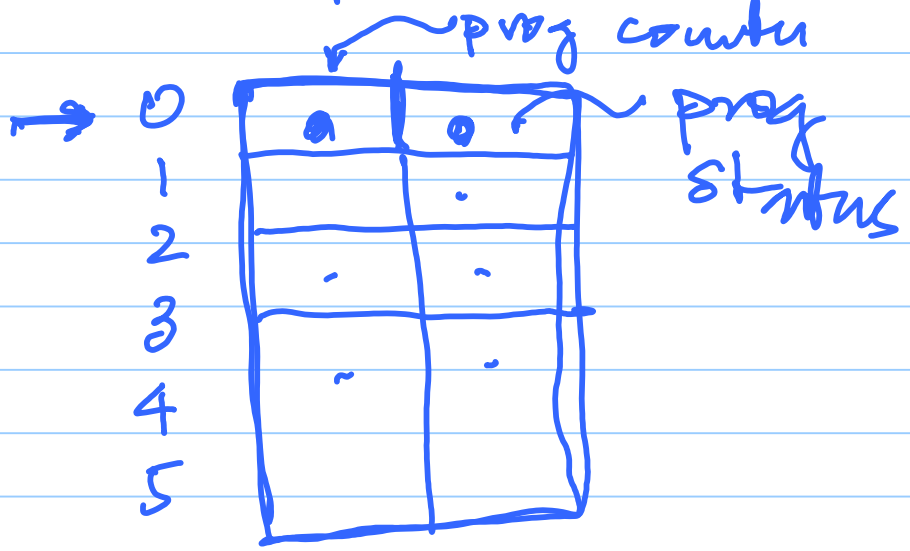
priv calls


interrupts
CPU
number
0
1
2
3
be interrupted
wire
0
1

Interrupt types (numbers)

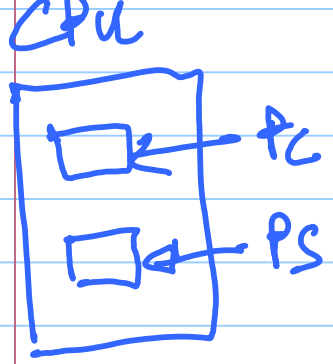
0, 1, 2, 3

Interrupt vector



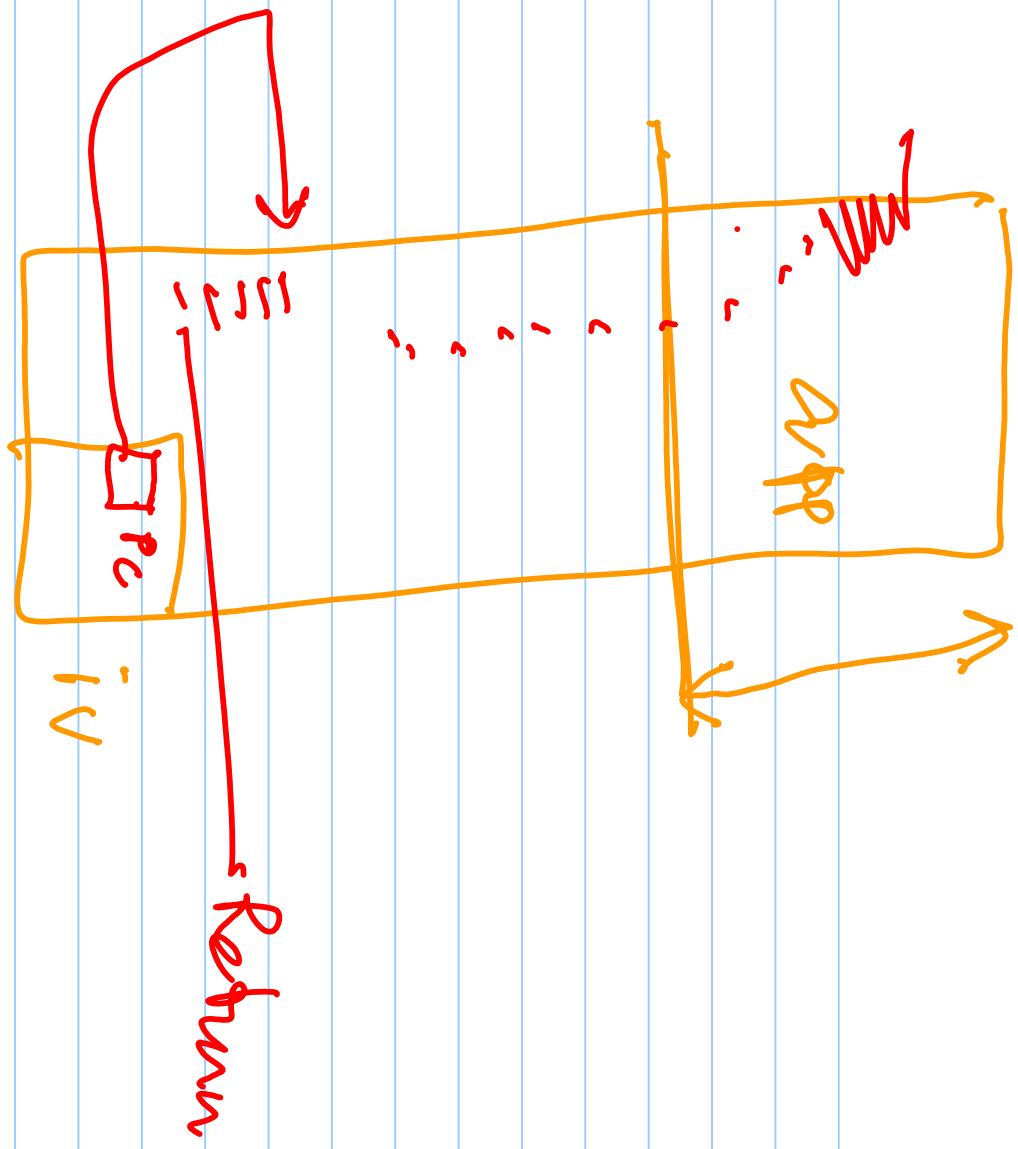
fixed part of memory

CPU gets an INT of type i



microcode

- push PC — copy PC to memm location pointed by SP ; dec SP
- push PS
- load PC ← Vector[i].PC
- " PS ← vector[i].PS
- ↓continue


pc
Return

PS, aka flags – prog status

SP → stack ptr

PC → IP → prog counter