CSE 330: Operating Systems

Spring 2018

Class: 28 **Date:** 4/24

Note Title

PART 1

PART 1

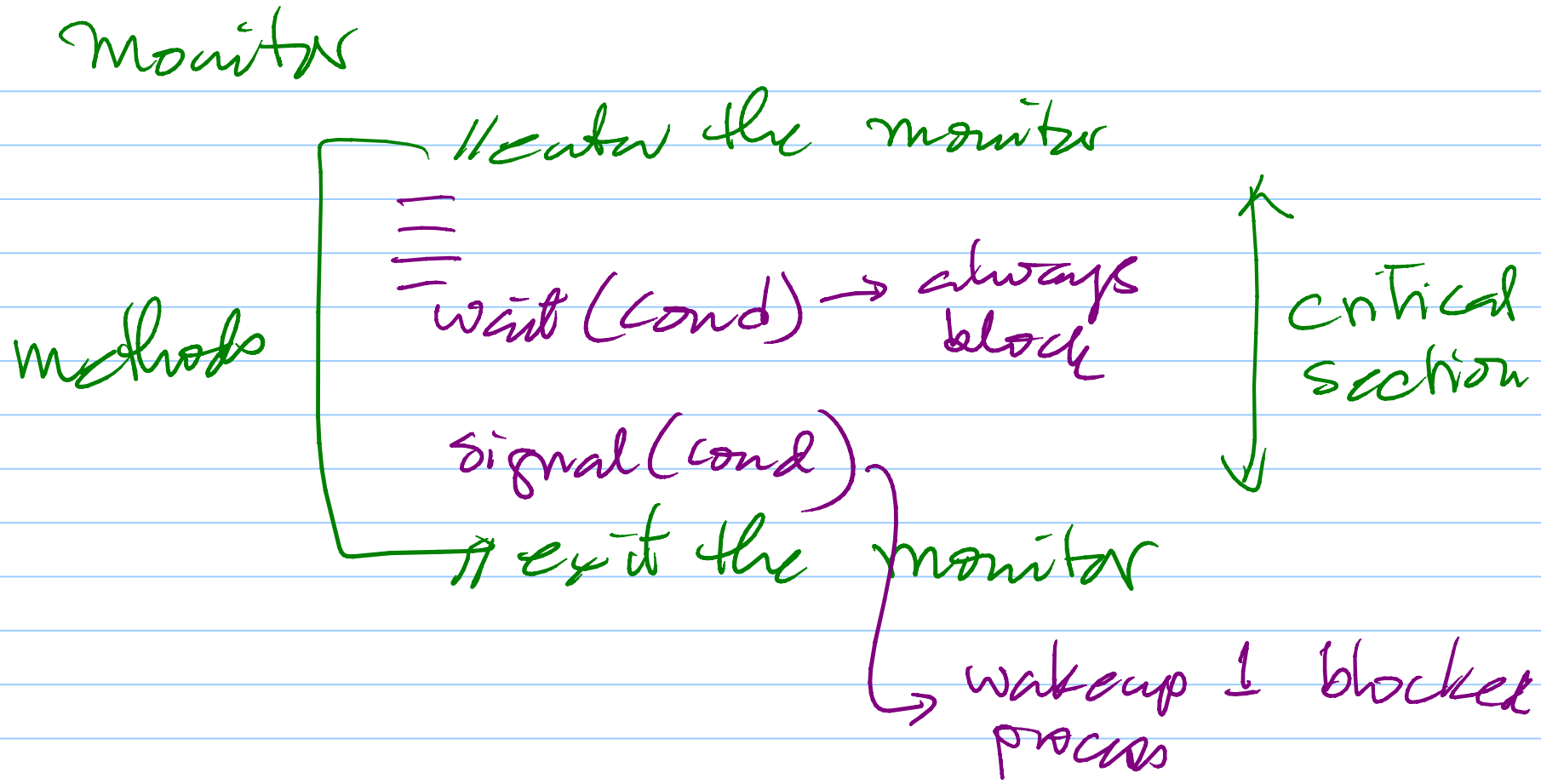
} 60 - 70 min.

Semaphores

init (S, value)

P(S) → wait if 0
decrement S (atomic)

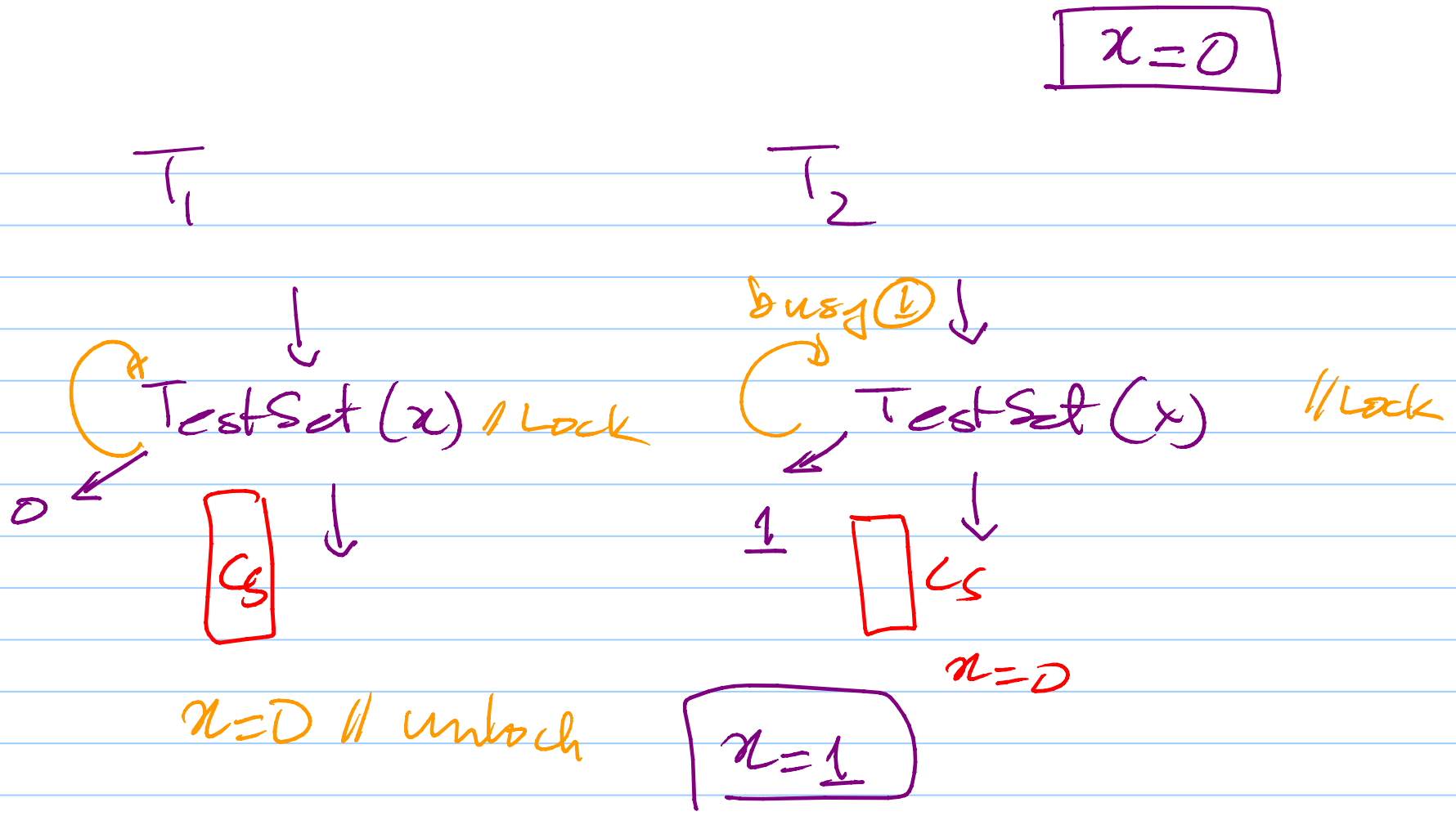
V(S) → increment (atomic)
wakeup

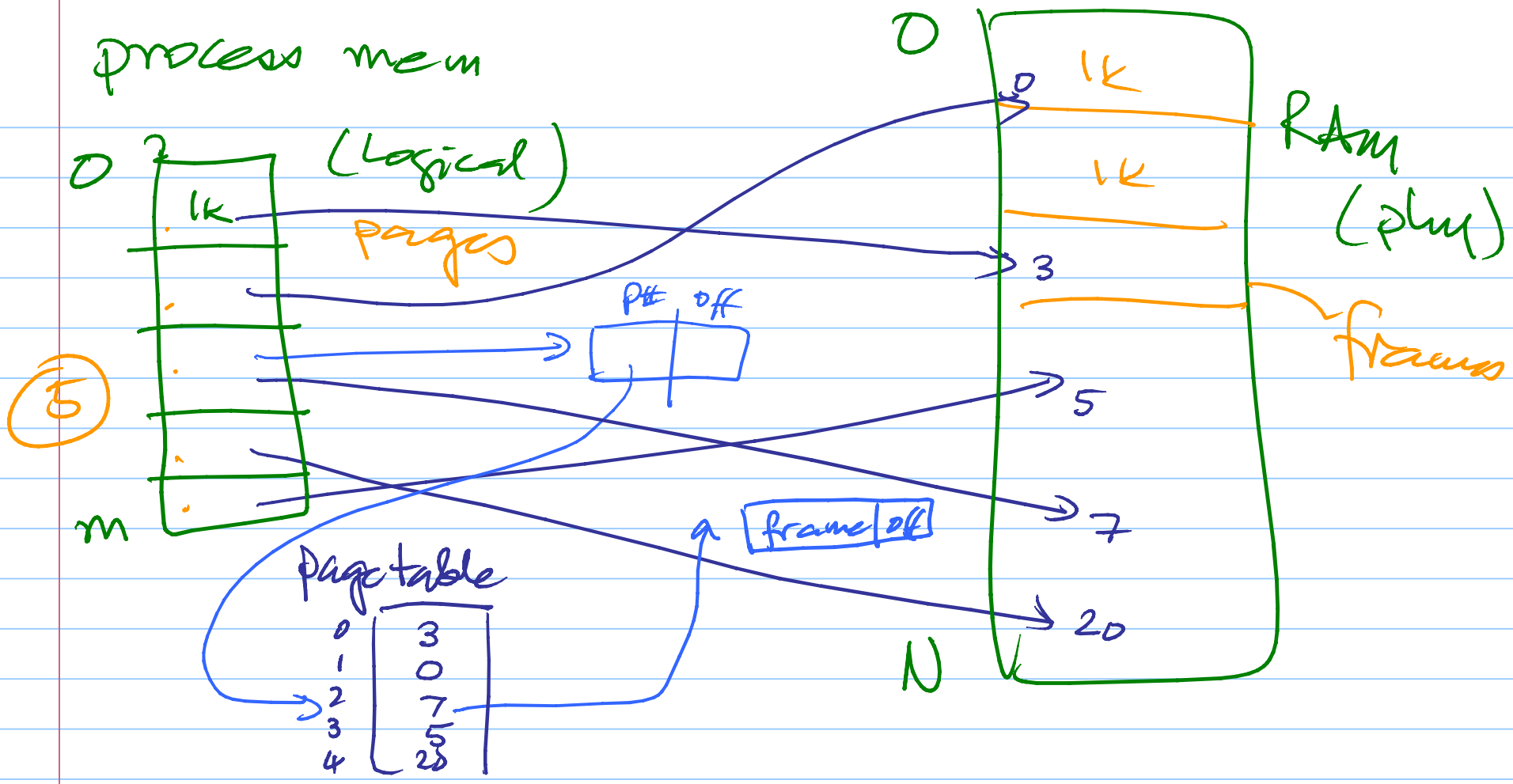

Monitor
methods
// enter the monitor
wait (cond) → always block
critical section
signal (cond)
// exit the monitor
wakeup 1 blocked process

test & set → one opcode
one parameter (addr)

Test & set (x) → [x will get value 1
prior value of x will be in a register (private)

Atomic


x=0
T1
T2
busy
1
TestSet(x) // Lock
TestSet(x) // Lock
0
1
CS
CS
x=0 // unlock
x=0
x=1


process mem
(Logical)
0
1k
pages
m
5
page table
0 3
1 0
2 7
3 5
4 20
pg# off
frame off
0
1k
1k
RAM
(phy)
frame
3
5
7
20
N

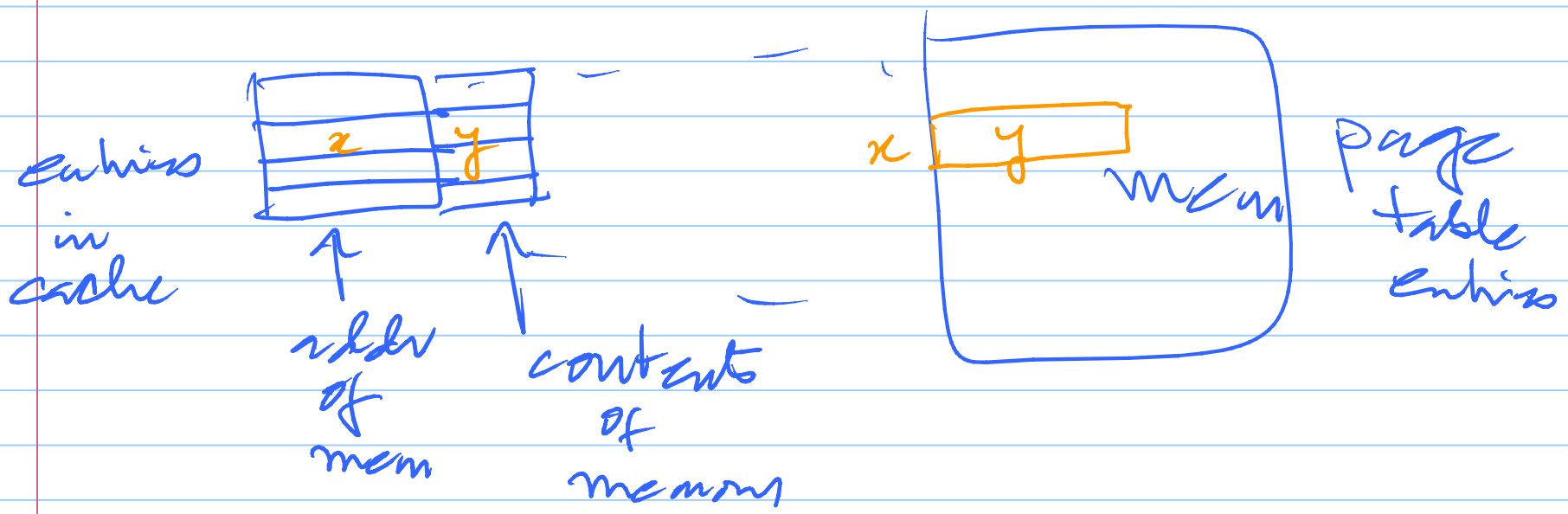
LAZY, pure, demand paging

↳ no pages in mem @ start

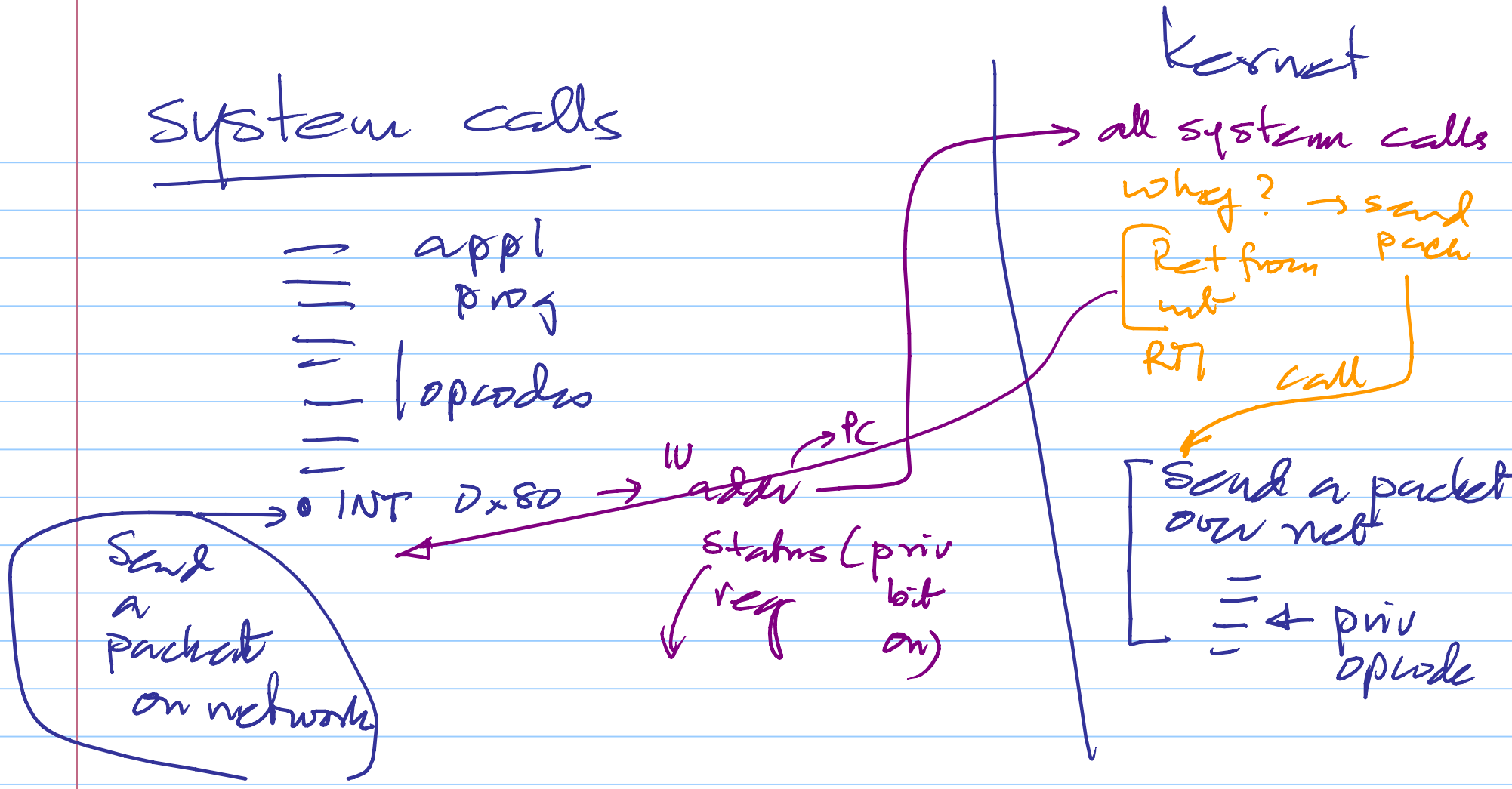
↳ do not swap pages out till mem is full

↳ finally will need a page out for every page_in

TLB – page table caching



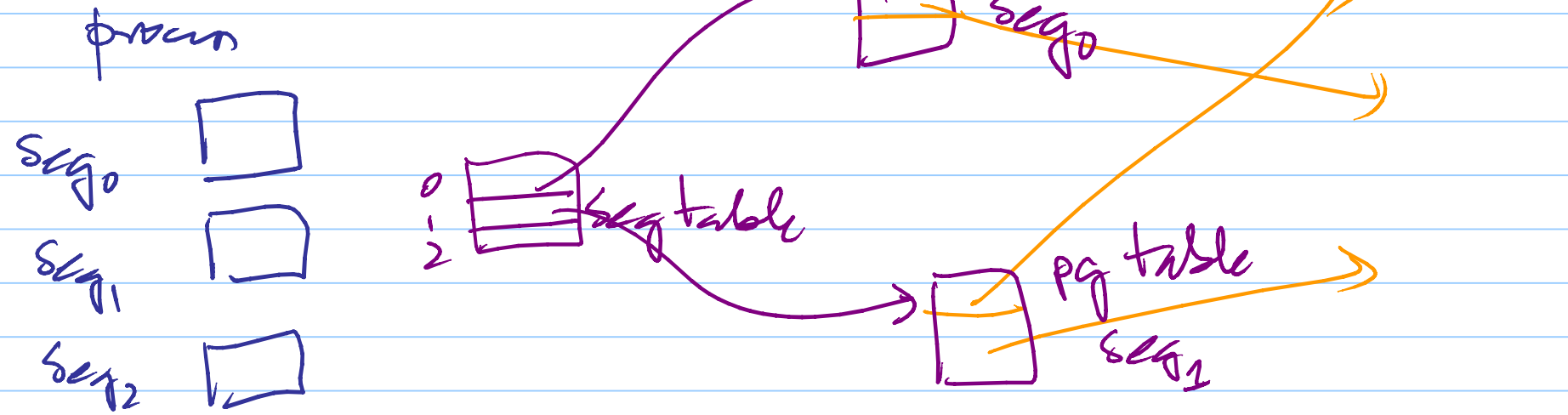
- Swapping → all contents of a process is either in mem or not
  - whole process in mem?


System calls
appl prog
opcodes
INT 0x80
IV
addr
PC
Send a packet on network
status (priv req bit on)
Kernel
all system calls
why? → send pack
Ret from int
RTI
call
Send a packet own net
priv opcode

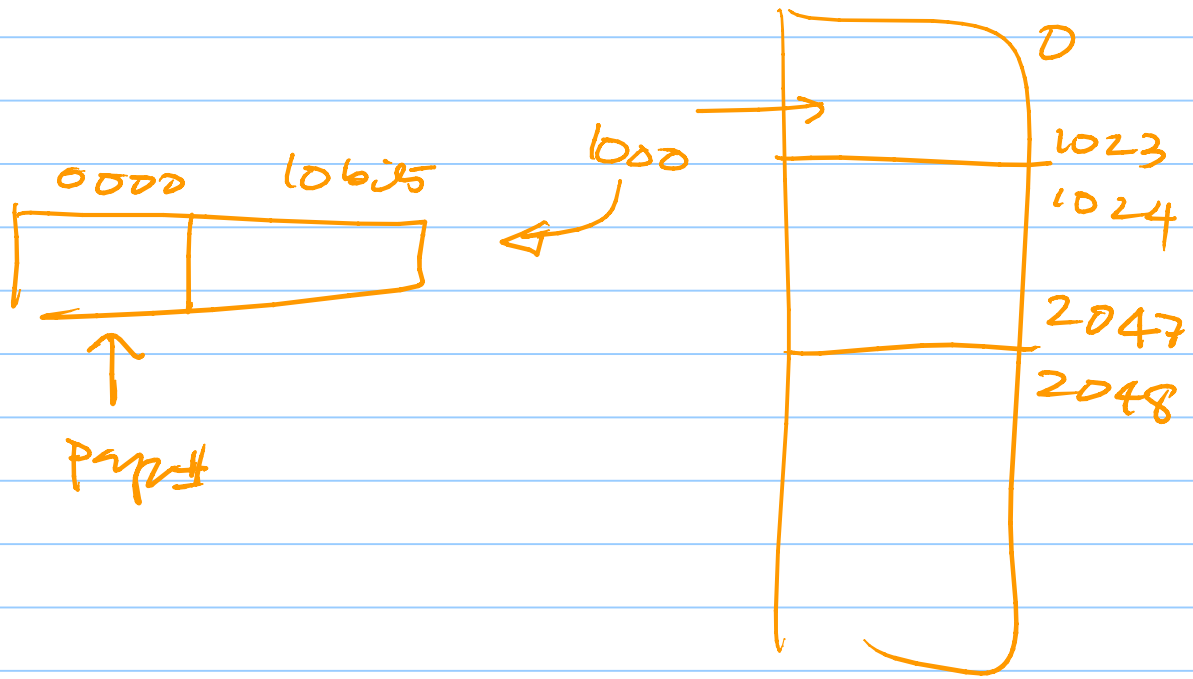
Paged Segmentation

↓

process memory is segmented



1 kbyte pages → 1024



fix deadlocks

- ① → write deadlock free programs (prevention)
- ② → resource allocation → safe states eg Banker's Alg
- ③ fix deadlocks if they happen

→ look for deadlocks

↳ detect (via WFG or RAG)

↳ NO deadlock →

↳ YES

↳ kill one process that is involved