CSE 330: Operating Systems

Class: 23 **Date:** 11/10

Fall 2016

Note Title

Deadlock detection

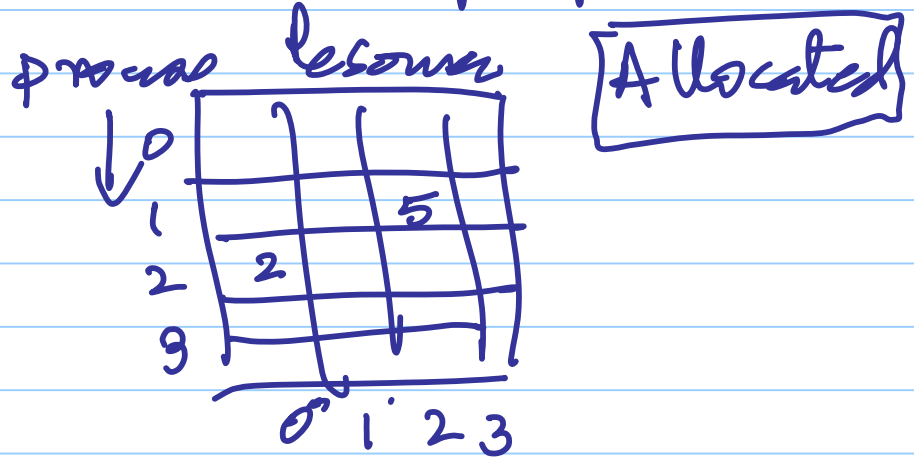
- single instance of each resource
- WFG
  - when an edge is added check for a cycle...

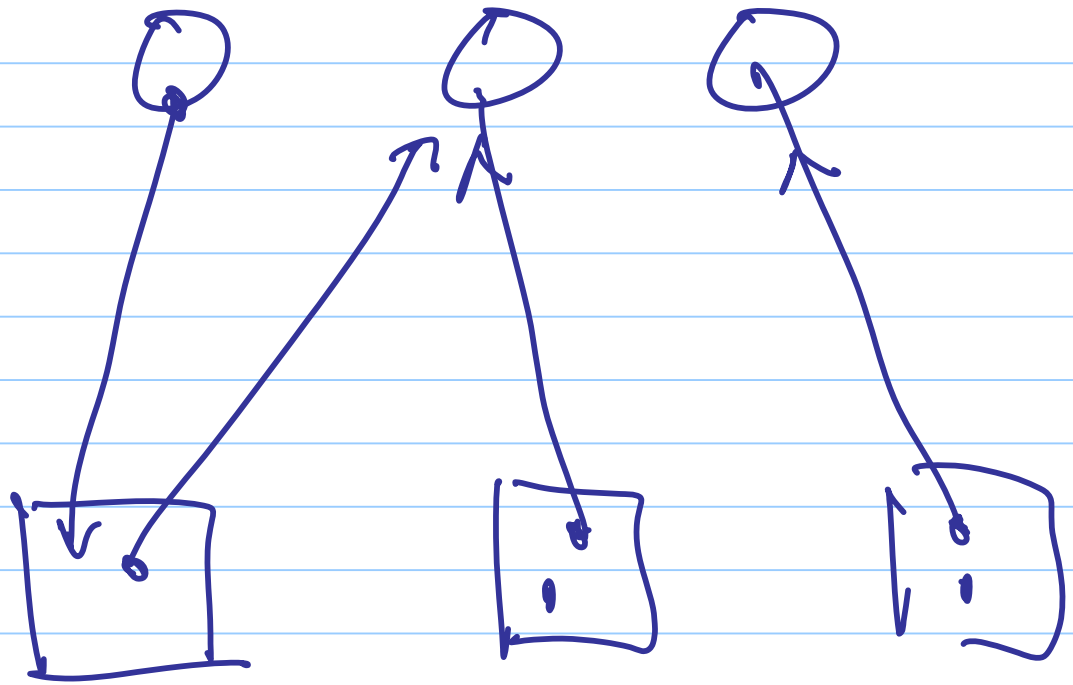
- Multiple instances of each resource

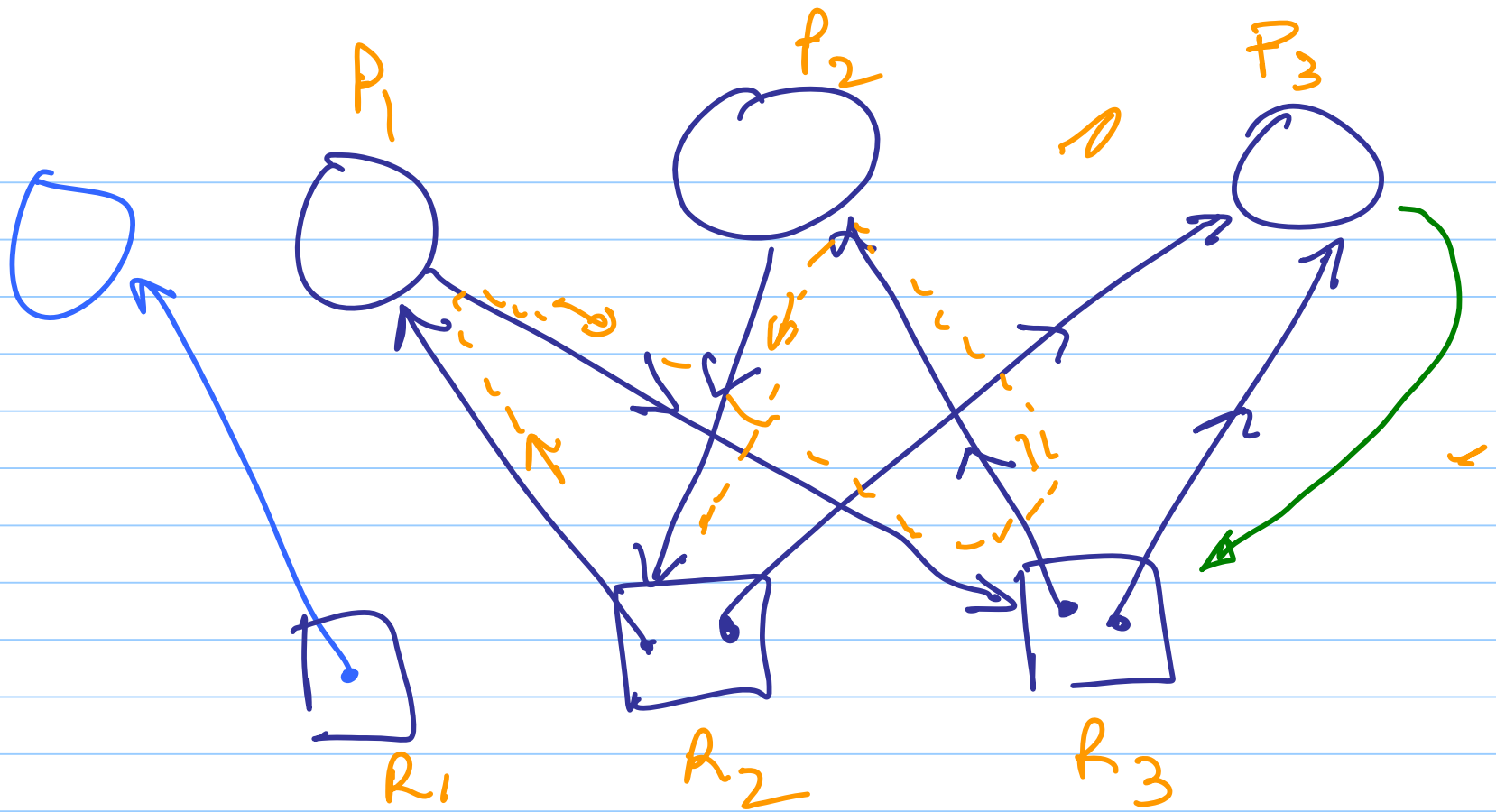
  - Resource Allocation Graph (RAG)

OR

TABLE

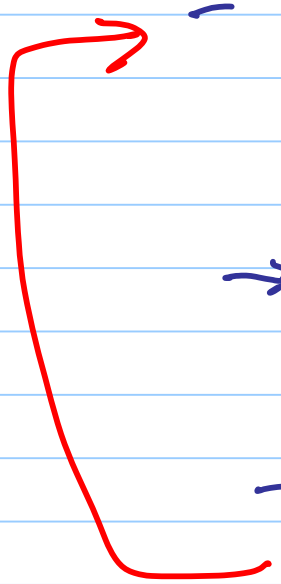


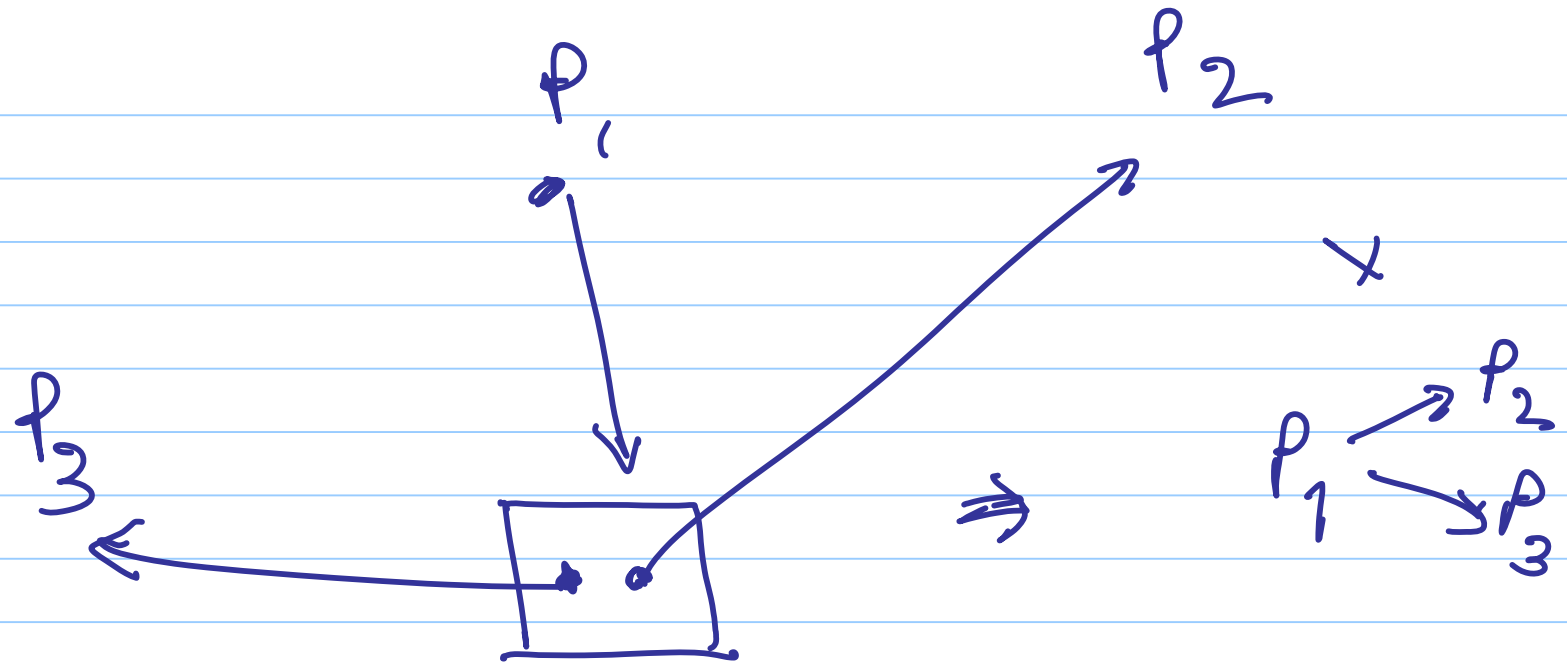


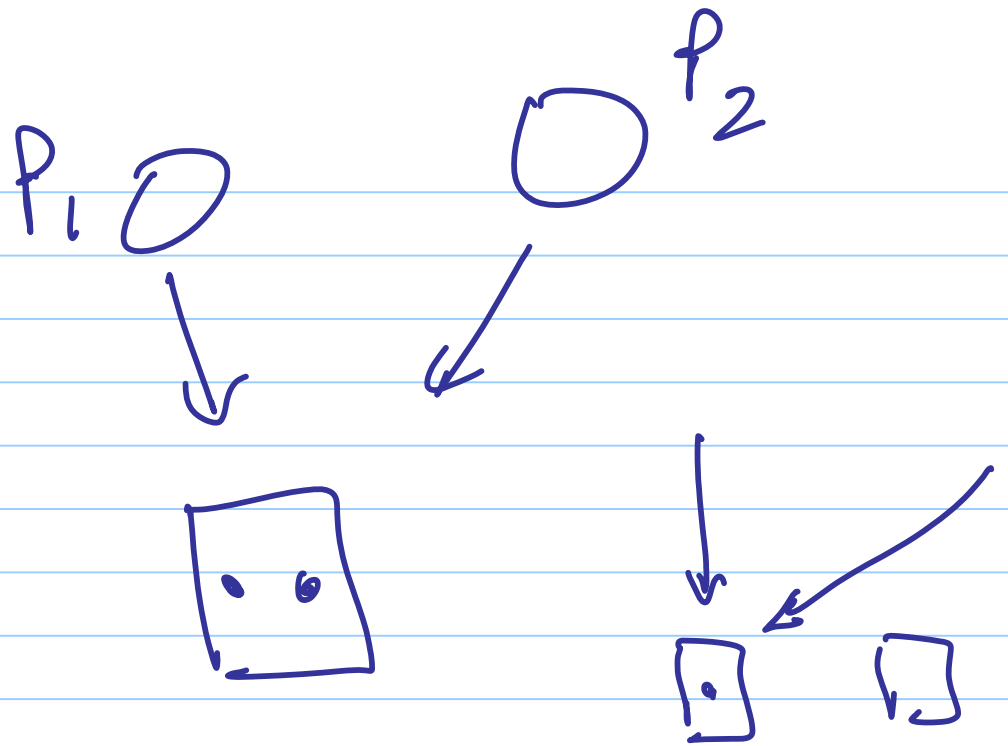

P1
P2
P3
R1
R2
R3

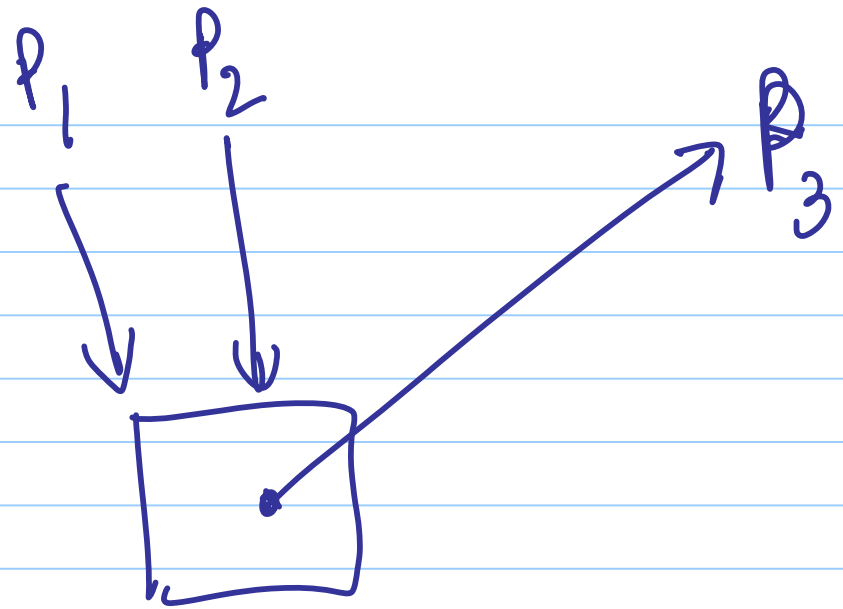
Deadlock detection

- make a temp copy of the RAG
- find a process with no outgoing edges → P_i → no such process → deadlock
  → delete P_i & free up resources allocated to P_i
  → Allocate the new resources → to requesters




p_1
p_2
p_3
X
p_1
p_2
p_3


p_1
p_2

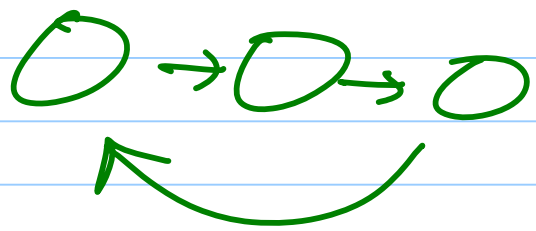

P_1
P_2
P_3

IF deadlock

→ find a set of processes (all have outgoing edges)

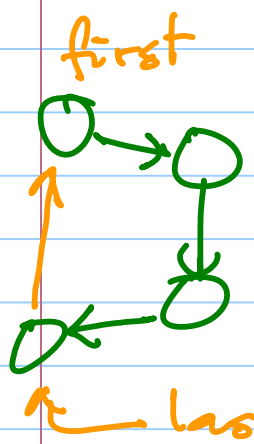
→ those are involved in the deadlock

Deadlock recovery

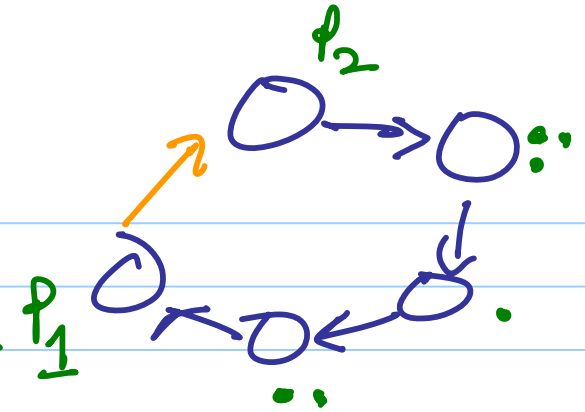


- terminate (something?)
  - free up resources, restart

- Terminate all processes in cycle
  - ↳ overkill
- Terminate 1 process
  - → the "last" process that caused the cycle
  - → the "first" process that was involved.



- Last one
- first one



- one with the largest # of resources held
- one with the smallest ...

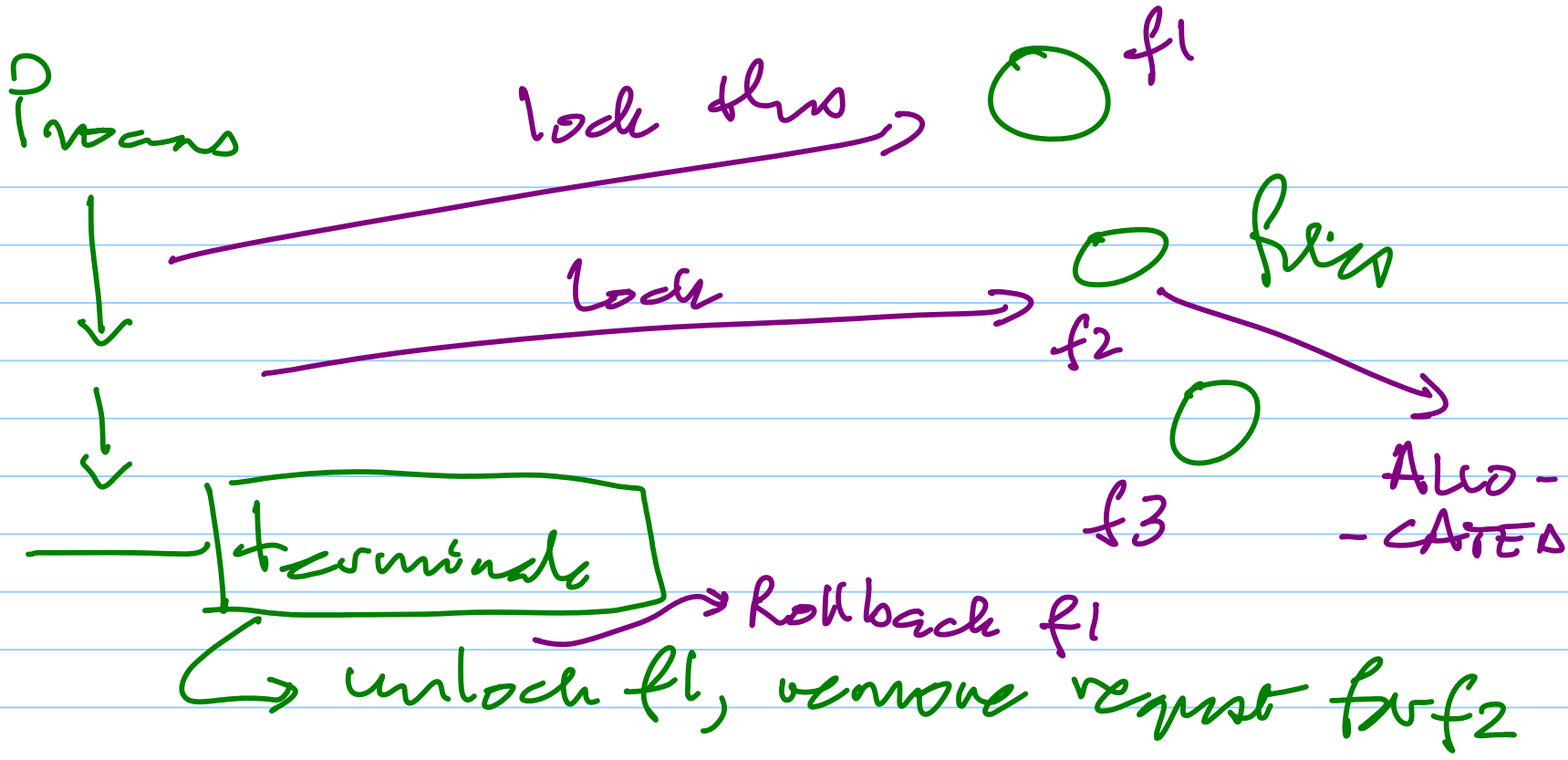
Process gets restarted ...

... then what?

what if it creates another deadlock?

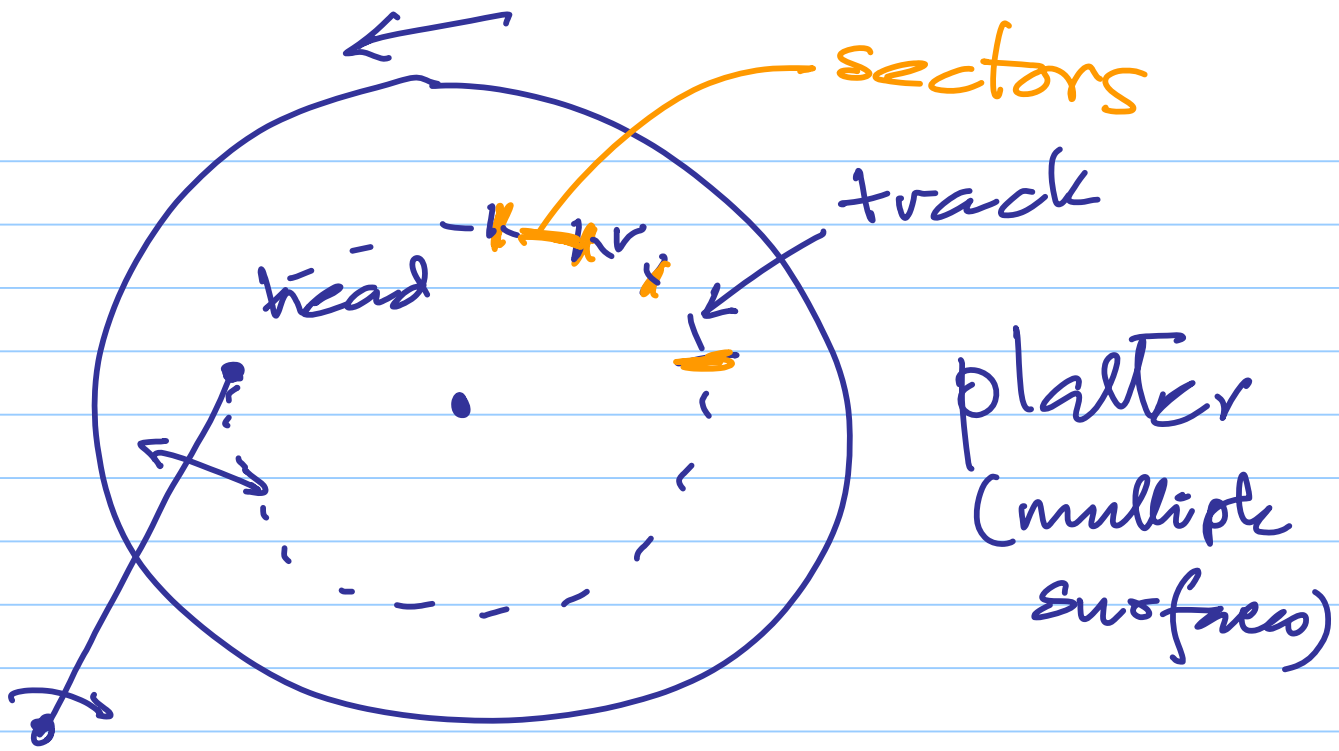
→ can lead to starvation?

⇒ YES


Process
lock this
f1
lock
f2
free
f3
Allo-
-cated
terminate
Rollback f1
unlock f1, remove request for f2

Storage devices

- magnetic disks
- solid state disks

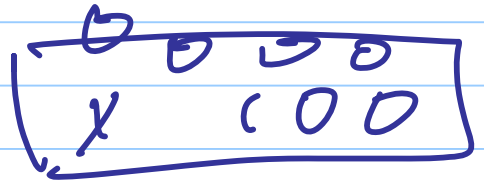

sectors
track
head
platter
(multiple
surfaces)

SSD

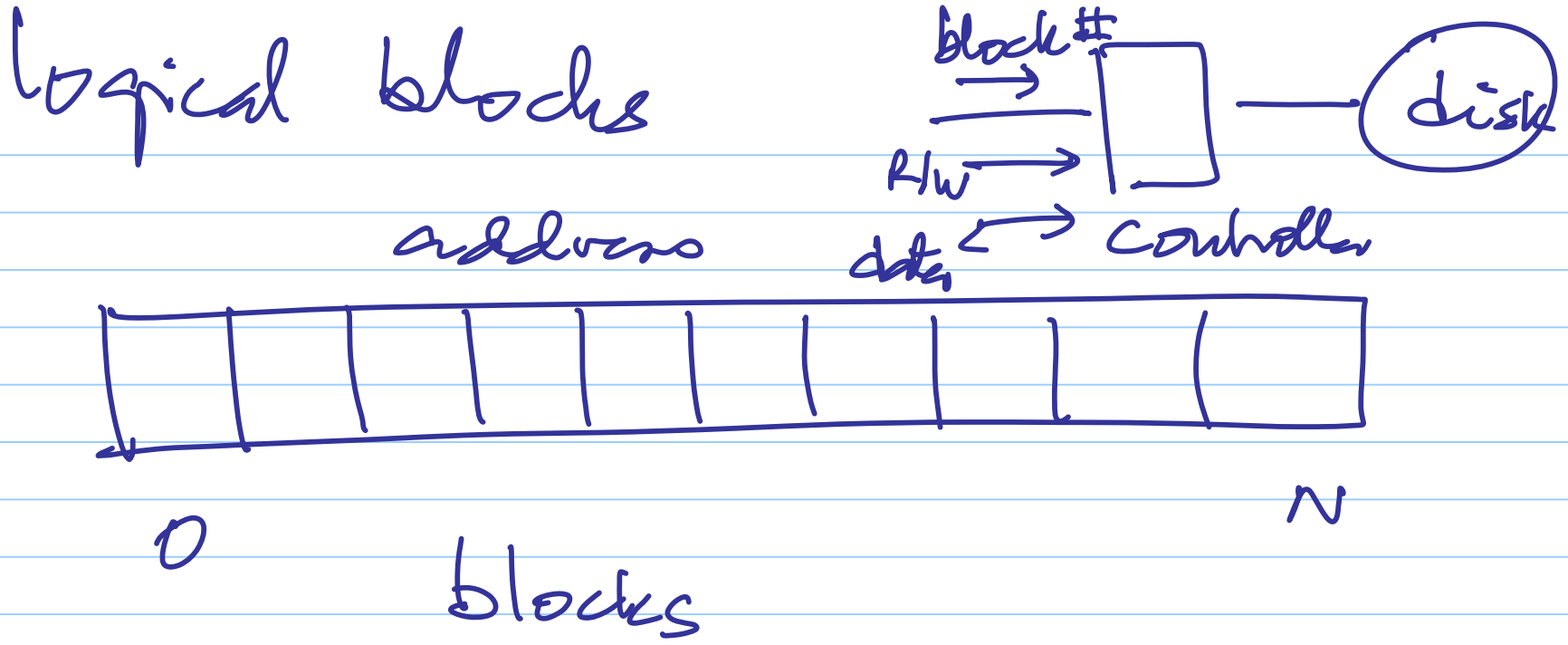
- no moving parts
- similar to RAM — but non volatile
- much faster than magnetic
  - WRITE is order of magnitude slower than ~~a~~ READ

Write

- erase
- write

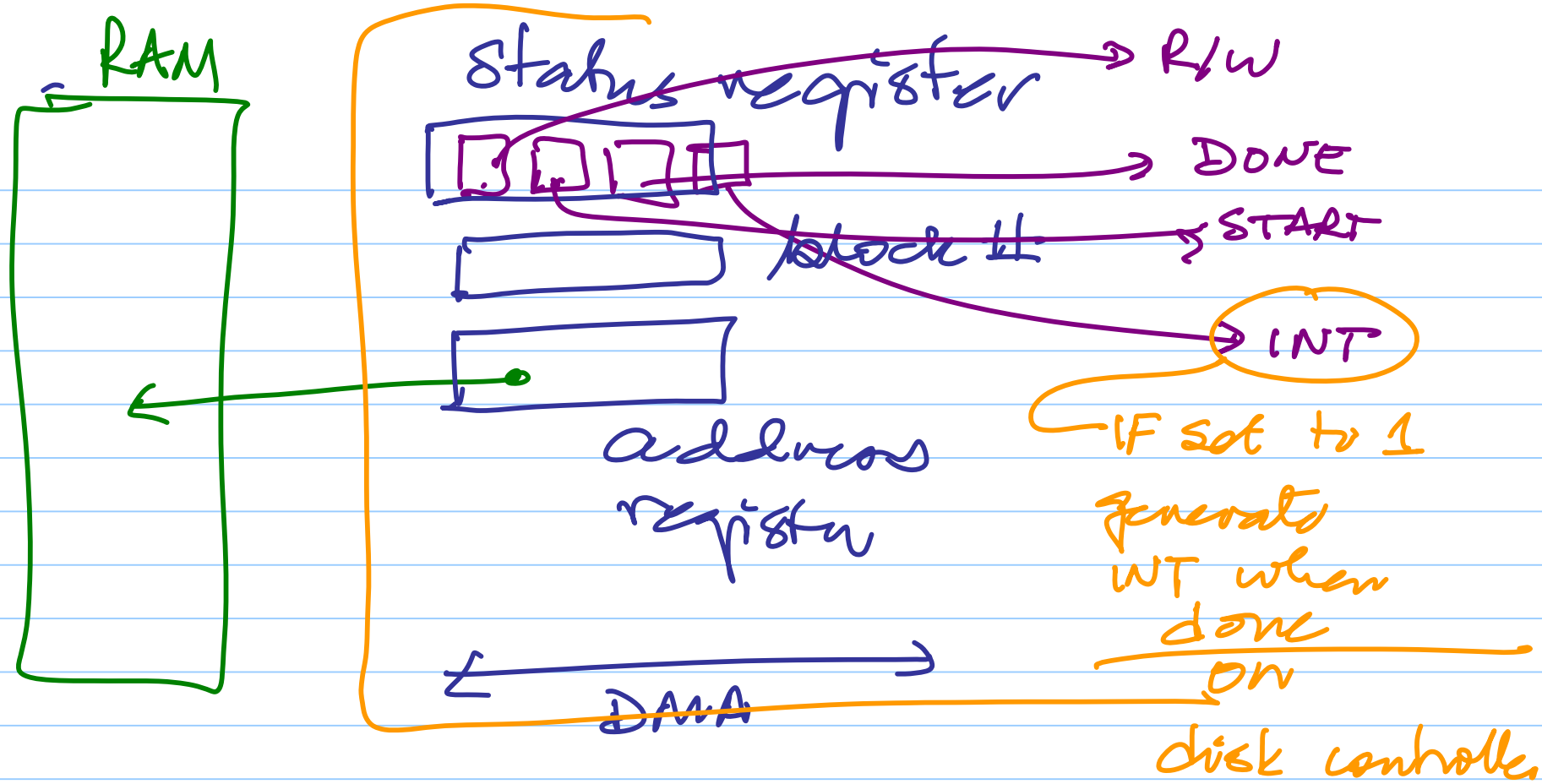


→ destructive!



read & write by block #

- how to read/write from software?
  - talk to controller via registers & DMA


RAM
status register
R/W
DONE
START
block #
INT
IF set to 1
generate
INT when
done
on
disk controller
address
register
DMA