Processes Continued

- Context Switch ✓
  - what state to save + how
- Scheduling
  - i.e, when to context switch
  - preemptive, cooperative
- Interrupts, Exceptions
  - System calls ☆

- IPC

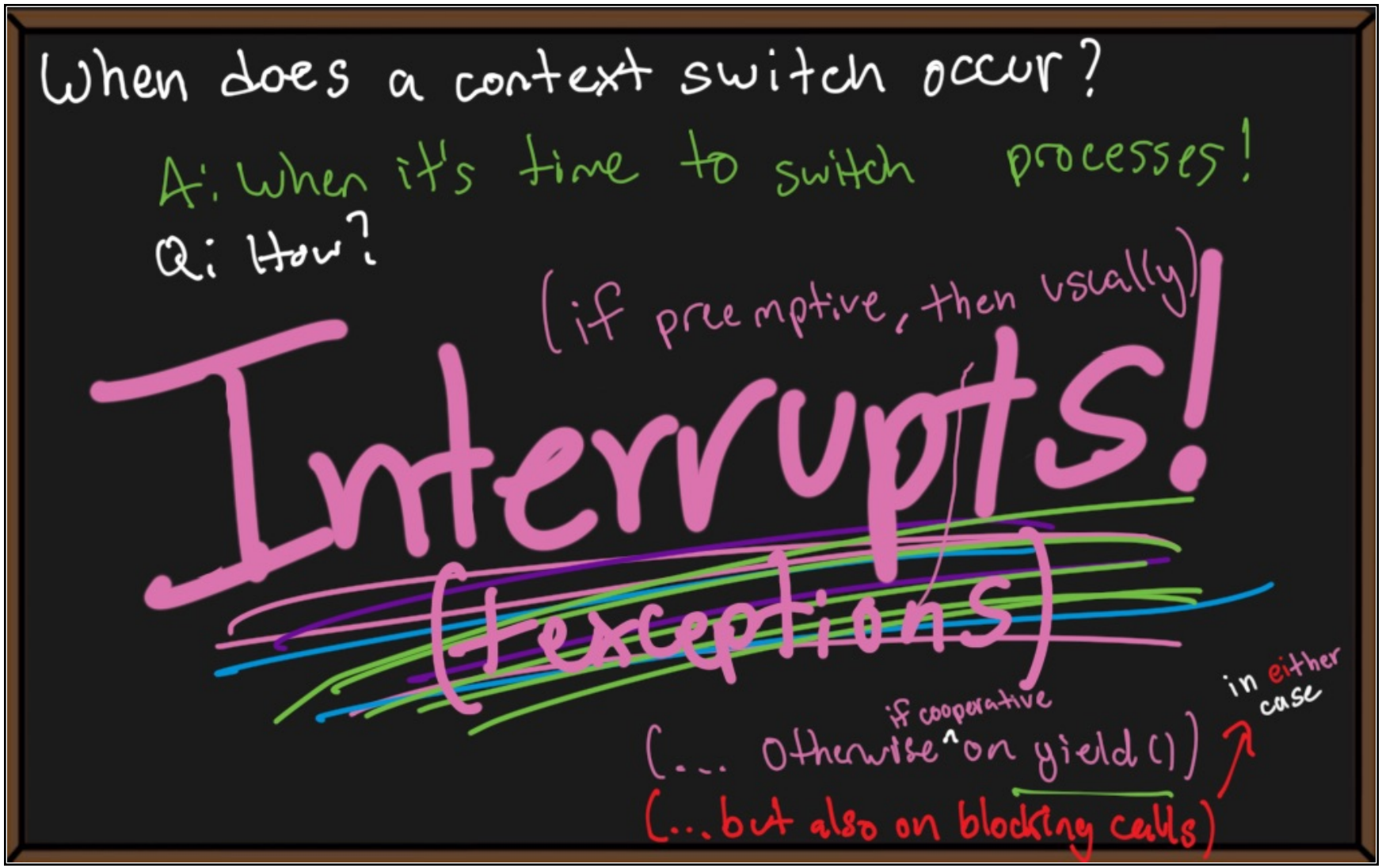
LAST LECTURE

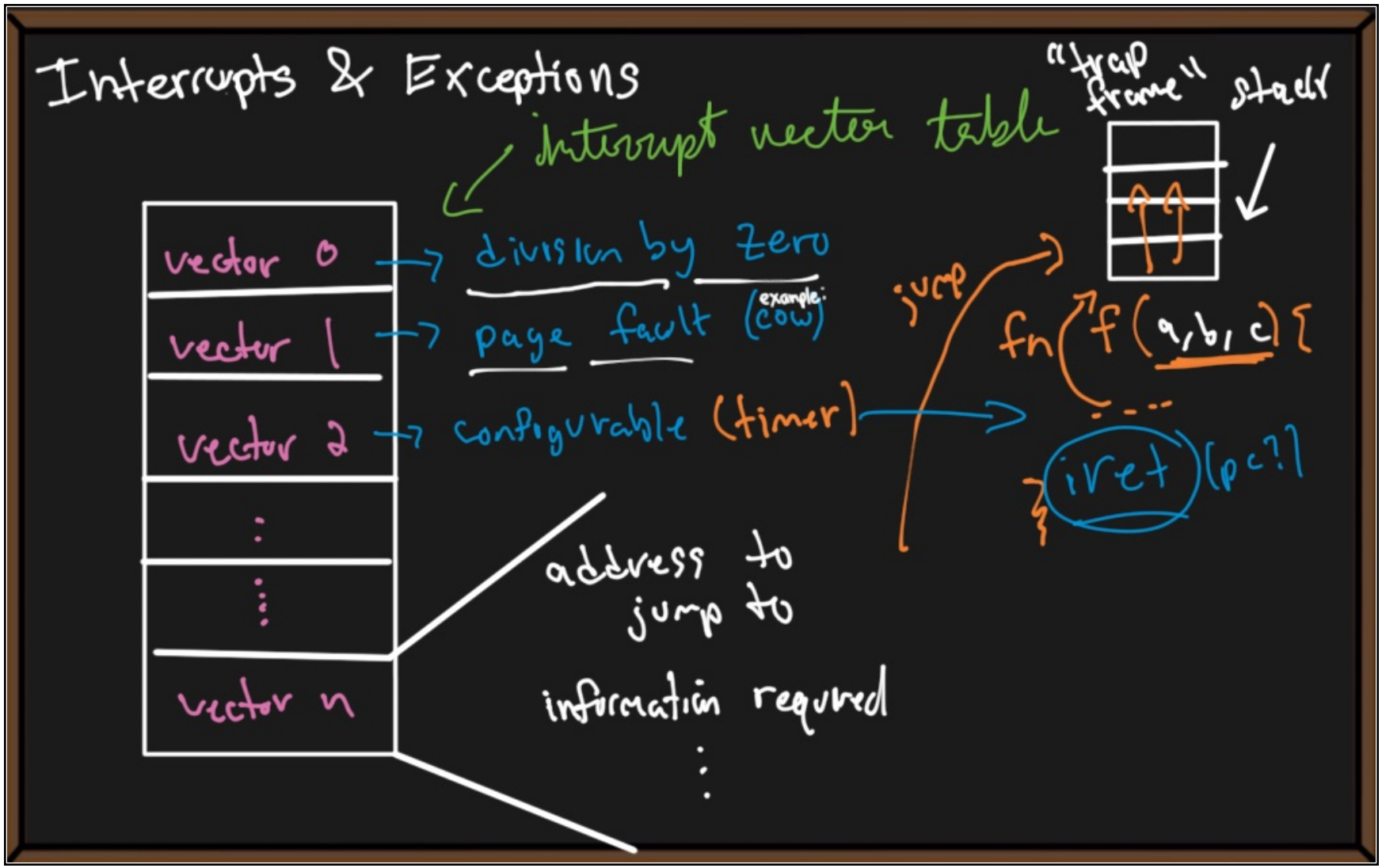
What state is needed to restore "virtual" machine?

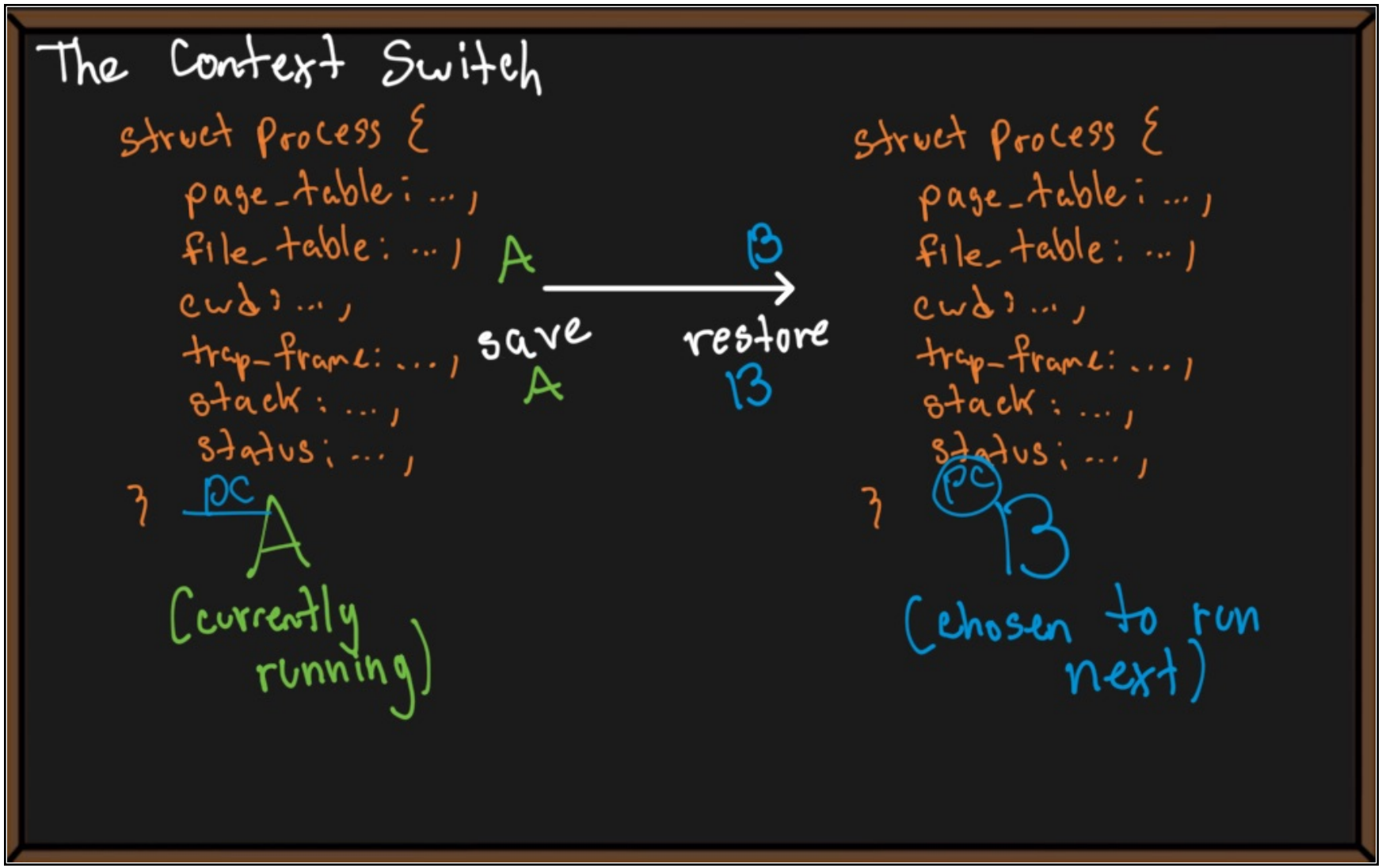
Machine Abstraction

- Memory
  - Isolated Machine
  + virtual memory
- Processor
  + time slicing
- Disk (global / share)
  + Network


Process State ('proc', 'task_struct', 'PCB')
1) for virtual memory?
- pointer to page table
Box<Page>
+ flush TLB (tags)
2) for file system?
- file table
- cwd Dir
Vec<FileInfo> or
[FileInfo; MAX_FILES]
3) for time slicing?
- 'trapframe' manual
- registers: data + other cpu context
TrapFrame
ready, waiting, zombie
- scheduler status
4) Kernel stack
*) metadata (pid, parent ...)


When does a context switch occur?
A: When it's time to switch processes!
Q: How?
(if preemptive, then usually)
Interrupts!
(+exceptions)
(... otherwise if cooperative on yield())
(...but also on blocking calls)
in either case


Interrupts & Exceptions
interrupt vector table
"trap frame"
stack
vector 0
vector 1
vector 2
vector n
division by zero
page fault (example: COW)
configurable (timer)
address to jump to
information required
jump
fn
f(a,b,c){
iret (pc?)


The Context Switch
struct process {
page_table: ...,
file_table: ...,
cwd: ...,
trap_frame: ...,
stack: ...,
status: ...,
}
PC
A
(currently running)
A
B
save
A
restore
B
struct process {
page_table: ...,
file_table: ...,
cwd: ...,
trap_frame: ...,
stack: ...,
status: ...,
}
PC
B
(chosen to run next)

Scheduling

Which process (in general: "thing") do I run next?

Where do I run the next ~~process?~~ thing?

- May have multiple CPUs.
- CPUs may be heterogeneous.
- May even want to schedule on not CPUs.
  i.e. GPU, accelerators, etc.