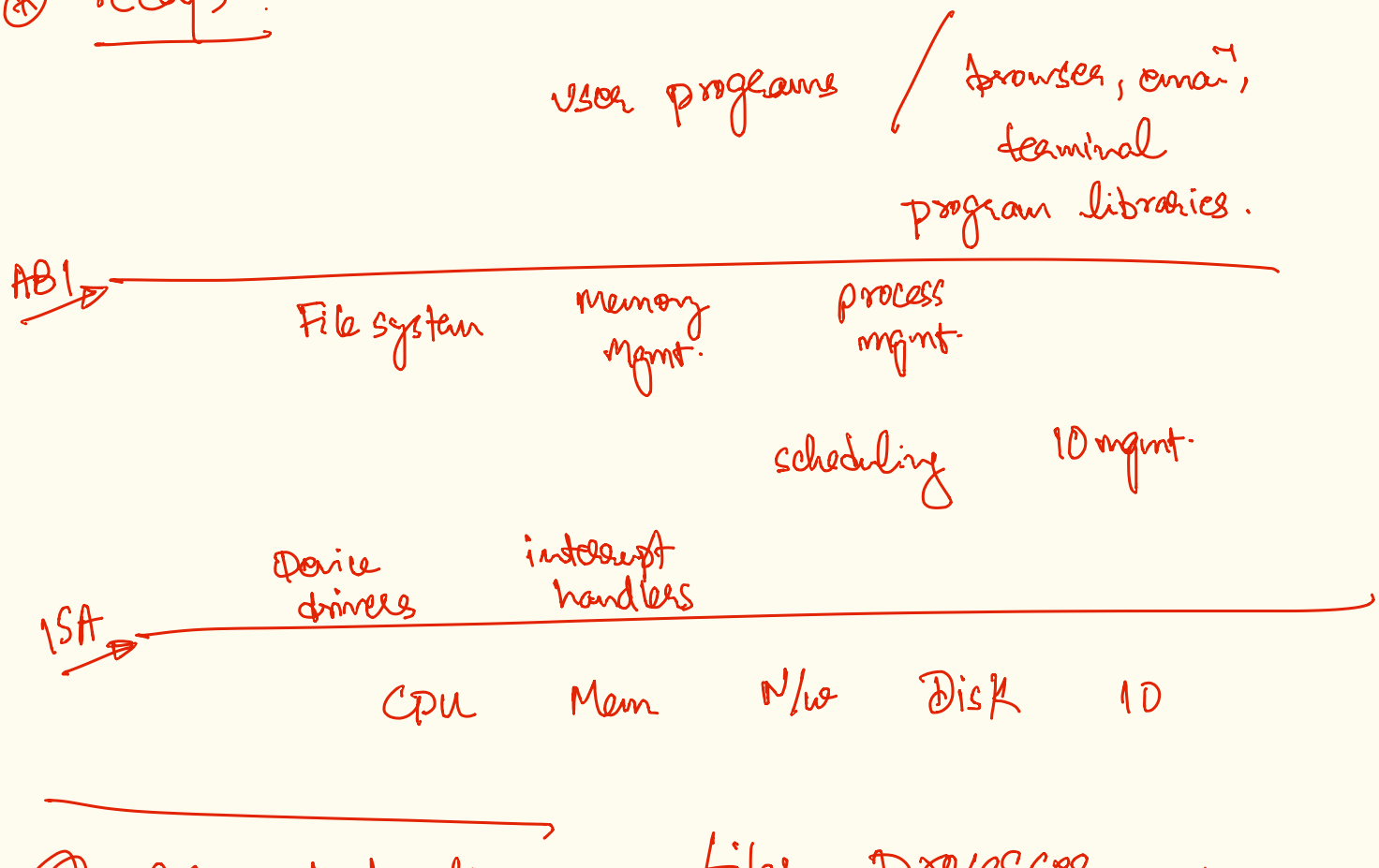
Lecture 3 CS 333/347 4.08.2022

⊛ recap:



⊛ OS abstractions — files, processes ...

OS requirements: multiplexing / sharing
isolation
robust
uniform interface
fairness

Key two building blocks:

(i) privileged modes of execution

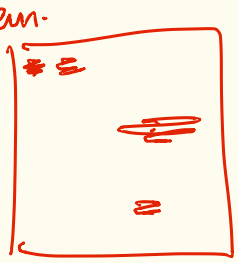
(ii) interrupts / interrupt-driven execution.

(i) privileged modes.

⇒ OS is the "owner" of all resources. ← CPU, memory, IO...

⇒ to prevent "bad" things from happening.

- programs accessing each other's memory.
- monopolizing the CPU
- all packets on NIC to a single processes
- crash of one app. leading to m/c meltdown.



(ii) (privileged) modes of execution (on the CPU)

user mode (less privileged) — user programs

user program execution

Kernel mode (more privileged) — OS

- setup/allocate mem.
- pick next program to execute
- handle exceptions...

HW

Kernel vs. OS vs. Ubuntu ?.

- OS executes in the highest PL

riding on hardware support.

- each instruction (of the ISA) has min. Privilege level for correct execution.
- w/o privileges ← nop, undefined.

(ii) interrupts.

- events of the world are non-deterministic!

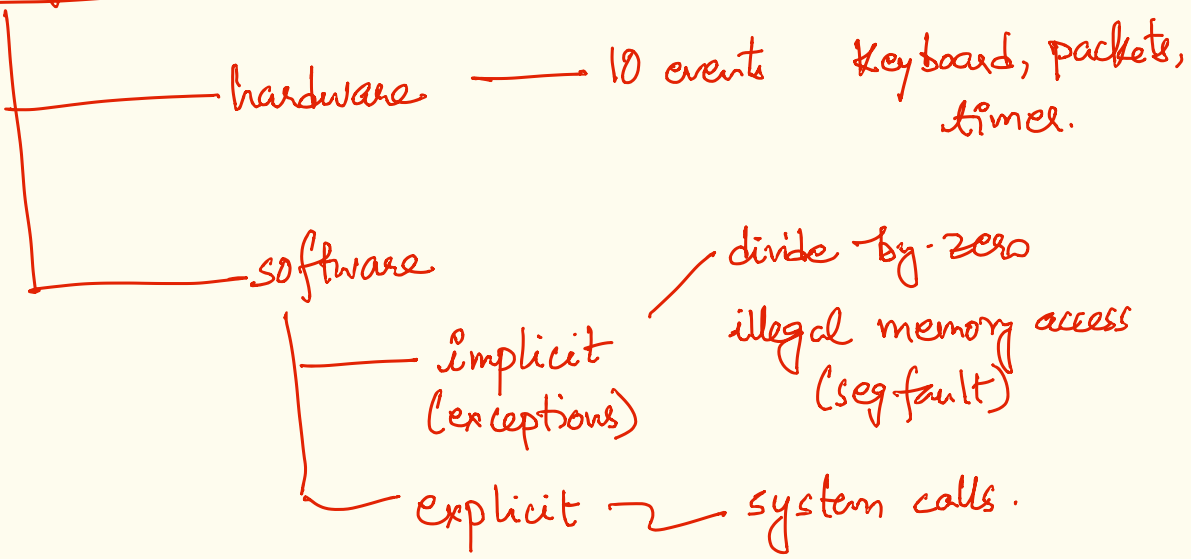
(i) all external events (input-output) do not follow a schedule.

e.g: arrival of network packets, key press on keyboard, read from disk etc.

* how does CPU know when to / which events to process these IO events?.
  - interrupts & polling.

(ii) basic mechanism to invoke OS services.

* categorization of interrupts



* what happens on an interrupt?.

- CPU pauses execution (after current instruction).
- saves information about current execution (program counter etc.)
- switches to Kernel mode (if required)
- jumps to interrupt service routine (ISR) code.
  - the interrupt handler is an OS component and provides control for OS for OS actions/control/ownership.

‖ interrupt itself is h/w state change, e.g: change of voltage level on a CPU pin.

↳ the interrupt handler.