Interrupts

1. What are interrupts ✓
2. Hardware (asynchronous) vs Software (synchronous) interrupts ✓
3. How interrupts works in OS ✓
4. Interrupt handlers ✓
5. Hardware interrupts vs Software interrupts vs Signals. (Exceptions)

What is an Interrupt

- An important responsibility of the OS is to manage hardware devices connected to machine.
- The OS needs to communicate with the machine's individual devices.

But there is an issue here . . .

- Processor speed >> hardware speed
- Not ideal to issue an request to a hardware device and then wait.
- Better to be free and handle some other work for the processor.

Ways to overcome the above issue:

t2
t2
t3
t4

Polling: The kernel can check the status of the hardware periodically.

- Limitation: Overhead of checking regularly whether the hardware device is ready or not.

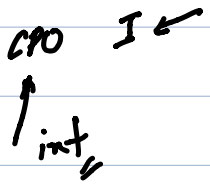
What if the hardware device has the ability to signal to the kernel when attention is needed?

Interrupt: An interrupt is an input signal to the processor indicating an event needs immediate action.

- Interrupts can either be generated in response to hardware or software events

Types of Interrupts:

Hardware or asynchronous interrupts



- Hardware interrupt is issued by a hardware device like Keyboard, disks, mouse etc.
- These interrupts can arrive at any time. Even when the processor is in middle of executing an instruction.
- Therefore, hardware interrupts are also called asynchronous interrupts.

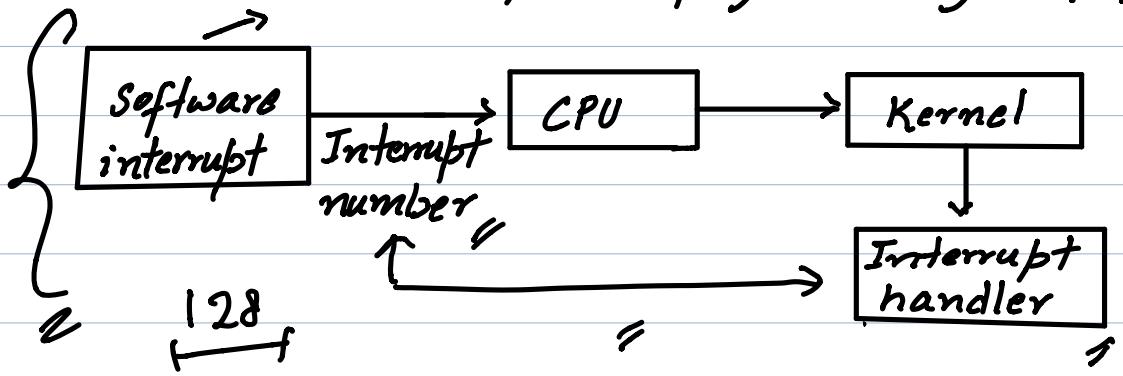
Software or synchronous interrupts (Exceptions)

- A software interrupt is issued by the processor itself.
- These interrupts are caused by the program that is running on the processor.
- These interrupts happen due to an instruction itself.
- Therefore, these interrupts are also called Synchronous interrupts (interrupts are in sync with the processor)

Types of software interrupts

Traps → system call

- Generated intentionally by the program.
- when a program wants to request an OS service.
- System calls are called from a program using traps.



Faults

- The program causes a recoverable error.
- Interrupt handler is called which handles the error.
- Examples: Segmentation fault, page fault

Aborts

- The program causes a non-recoverable error.
- The interrupt handler aborts the program.
- One cannot catch these types of errors.
- Example: Hardware errors (corruption of disk)

Classify the following into types of interrupts

- I/O Request (read/write) → Software / Traps
- Request more heap memory → " / Traps

- Division by Zero error ⟶ " / Fault "
- User presses key on keyboard → Hardware
- Disk controller finishes reading data → Hardware
- Context switch → Software ✓
  → Hardware ✓

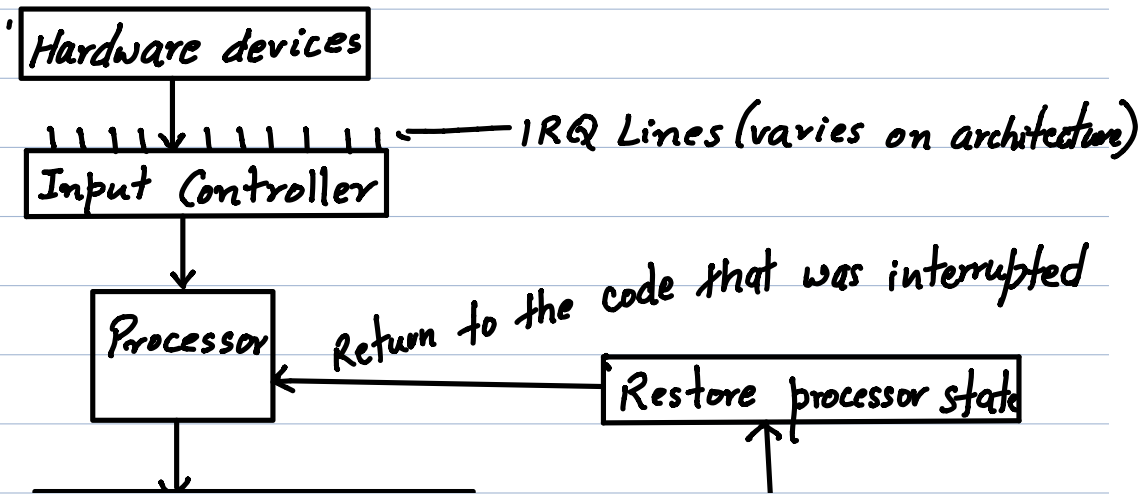
10 ms Timers

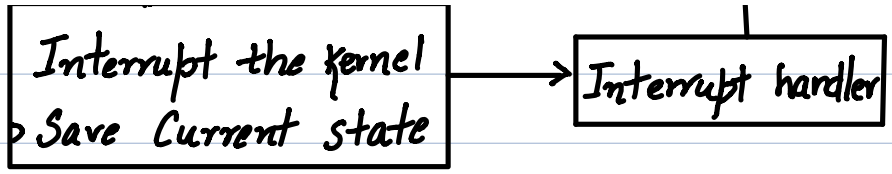
In next video :

- How does a program get input from the keyboard ?
- How does a program get data from the disk ?

Exceptions in OS ≠ Exceptions in Java or python

How Interrupt Works





- An interrupt is produced by signals originating from hardware devices.
- These signals are directed into input pins of input controller. The output is connected to the processor.
- After executing each instruction, the processor checks for the signal from the input control.
- If there is an interrupt, the processor interrupts its current execution and notifies the kernel.
- The kernel saves the state of current process and runs an interrupt handler in response to an interrupt.
- Each device that generates interrupts is associated with an interrupt handler.
- The interrupt handler determines the cause of interrupt, performs necessary processing, performs a state restore and returns the CPU to its prior state.

Interrupt handler → Interrupt Service Routine (ISR)

- While the processor is busy with an interrupt, the

interrupt system is disabled on the processor.

- Since the interrupt handler has interrupted other code and also possibly some other interrupts, they should be simple and quick.

Interrupt vector

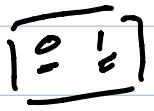
- The mapping between the interrupt handlers and their memory addresses is stored in Interrupt vector.

Interrupt handler number ⟶ Address

- Prevents the need for an interrupt handler to check all interrupts and determine which one it can serve.

Maskable vs Nonmaskable Interrupt

Maskable Interrupt



- These are non-critical interrupts that can be turned off by the CPU.
- Interrupts from devices (keyboard, mouse etc.)

Nonmaskable Interrupt

- These are critical interrupts that cannot be turned off.
- Eg. page fault, divide error, seg fault, etc.

Priority of Interrupts

- Each interrupt is also associated with a priority.
- Higher priority interrupts can preempt lower priority interrupts.
- When two or more devices interrupt processor at the same time, one with higher priority is given preference.

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Intel processor vector Table

0-31 → nonmaskable interrupts

32-255 → maskable interrupts