Tutorial 2: Computer Startup, System Organization, Operation, and Interrupts

1.What is a bootstrap program, and where is it stored?

The bootstrap program is a small piece of code responsible for starting the computer.
It is stored in **ROM (Read-Only Memory)** or **firmware**, and its main job is to load the operating system into memory and start it.

---

2. Why is the bootstrap program stored in ROM instead of RAM?

ROM is **non-volatile**, meaning its content remains intact even when the computer is turned off. Since the computer has no operating system loaded in RAM during startup, ROM ensures the startup code is always available.

---

3. Explain how the computer system operates when multiple components access memory simultaneously.

The system uses **scheduling and synchronization** mechanisms managed by the OS and hardware to ensure each device or CPU accesses memory in turn, avoiding interference.

---

4. What is an interrupt, and why is it important?

An **interrupt** is a signal that temporarily halts the CPU's current operation so that an **interrupt service routine (ISR)** can handle an event (like input/output completion or an error).
It allows the CPU to respond quickly to important or unexpected events.

---

5. Differentiate between hardware and software interrupts.

- **Hardware interrupt:** Triggered by devices (e.g., keyboard input, disk I/O completion).
- **Software interrupt (trap or exception):** Triggered by software, either due to errors or by explicit user/system requests.

---

6. What is an interrupt vector, and what is stored in it?

An **interrupt vector** is a table that contains the **addresses of all interrupt service routines (ISRs)**.
When an interrupt occurs, the CPU uses the interrupt vector to locate and execute the correct ISR.

---

7. Describe the steps involved in interrupt handling.

1. The CPU detects an interrupt signal.
2. The current **program counter (PC)** and **registers** are saved to preserve the CPU's state.
3. The CPU uses the **interrupt vector** to find the appropriate ISR.
4. The ISR executes to handle the event.
5. The CPU restores the saved state and resumes the interrupted process.

---

8. What is the role of the operating system in handling interrupts?

The OS manages interrupt handling by:

- Saving the CPU state.
- Determining the type of interrupt.
- Executing the correct ISR.
- Restoring the interrupted process afterward.
  This makes the OS **interrupt-driven**, meaning it responds to events as they occur.

---

9. Define trap or exception and give an example.

A **trap** or **exception** is a **software-generated interrupt**.
It can be caused by an error (like division by zero) or a user request (like a system call).

---

10. Why must the operating system save the CPU state before handling an interrupt?

To ensure that once the interrupt is processed, the CPU can **resume the interrupted program exactly where it left off**, without losing any data or instructions.