Table of Contents

[INTERRUPT 2](#_Toc129783070)

[WHY WE USE INTERRUPTS: 2](#_Toc129783071)

[ADVANTAGES: 2](#_Toc129783072)

[WITHOU INTERRUPTS: 2](#_Toc129783073)

[WITHOU INTERRUPTS: 2](#_Toc129783074)

[Interrupt request 3](#_Toc129783075)

[Interrupt Handler 3](#_Toc129783076)

[Interrupt Service Routine 3](#_Toc129783077)

[Types of Interrupts 4](#_Toc129783078)

[Addition of Interrupt cycle to Instruction cycle: 5](#_Toc129783079)

[THE INTERCONNECTION STRUCTURE: 7](#_Toc129783080)

[BUS INTERCONNECTION 7](#_Toc129783081)

[Bus Design 7](#_Toc129783082)

[Functional Groups of a Bus 7](#_Toc129783083)

[Bus Interconnection Scheme 8](#_Toc129783084)

[Operation of the Bus 8](#_Toc129783085)

[Types of Buses 8](#_Toc129783086)

# Chapter 03

# Top Level View of Computer

A computer system consists of three main components: CPU, memory, and I/O. Each component can have multiple modules. These components are connected together to perform the basic function of executing programs. We can describe a computer system by its external behavior, which includes the data and control signals exchanged between components. Additionally, we also need to understand the interconnection structure and the controls used to manage the use of the interconnection structure. INTERRUPT

Interrupt is a signal sent to the CPU by an external device, such as an I/O device or a timer, to request its attention. When an interrupt occurs, the CPU temporarily suspends the execution of the current program and switches to a special routine, called an interrupt handler, which handles the interrupt request.

## WHY WE USE INTERRUPTS:

We use interrupts to allow the CPU to handle multiple tasks concurrently, without requiring the CPU to constantly poll for events or wait for input. Interrupts provide a way for external devices to notify the CPU when they have data to be processed or require some action to be taken.

## ADVANTAGES:

Advantages of using interrupts include improved system responsiveness, efficient use of system resources, and the ability to handle asynchronous events. Interrupts allow the CPU to focus on executing the current program, while allowing external devices to operate concurrently, without waiting for the CPU's attention.

## WITHOU INTERRUPTS:

Without interrupts, the CPU would have to continuously poll for events or wait for input, which would be a waste of system resources and result in slower system performance.

## WITHOU INTERRUPTS:

With interrupts, the CPU can handle multiple tasks concurrently, allowing for more efficient use of system resources and faster system performance. Interrupts also enable the system to handle asynchronous events, such as I/O operations, without requiring the CPU to constantly check for events, freeing up the CPU to perform other tasks.

In summary, interrupts are signals sent to the CPU by external devices to request its attention, and are used to allow the CPU to handle multiple tasks concurrently, without requiring it to constantly poll for events or wait for input. Interrupts provide improved system responsiveness, efficient use of system resources, and the ability to handle asynchronous events. Without interrupts, the CPU would have to waste system resources and system performance would be slower. With interrupts, the CPU can handle multiple tasks concurrently, resulting in more efficient use of system resources and faster system performance.

## Interrupt request

Interrupt request (IRQ): Interrupt requests are signals sent by hardware devices to the CPU to request attention. IRQs are used to handle time-critical events and improve the efficiency of computer systems. Each hardware device is assigned a unique IRQ number, which is used to identify the device when an interrupt occurs.

## Interrupt Handler

Interrupt handler: Interrupt handlers are software routines that are responsible for managing interrupts. An interrupt handler is executed by the CPU in response to an interrupt request from a hardware device or software process. The interrupt handler is responsible for saving the state of the CPU, handling the interrupt, and restoring the CPU state when the interrupt has been handled.

## Interrupt Service Routine

Interrupt service routine (ISR): An interrupt service routine is a software routine that is called by an interrupt handler to handle a specific interrupt. An ISR is responsible for processing the data associated with the interrupt, such as data received from a keyboard or mouse, and updating the system accordingly. ISRs are typically short and efficient, as they are designed to handle time-critical events.

In summary, interrupt requests are signals sent by hardware devices to the CPU to request attention. Interrupt handlers are software routines that manage interrupts, while interrupt service routines are software routines called by interrupt handlers to handle specific interrupts. Together, these components enable the CPU to handle time-critical events efficiently and improve the overall performance of computer systems.

## Types of Interrupts

1. **Program Interrupts:** Program interrupts are generated by software when the CPU encounters an error or exception during program execution. Examples include division by zero, page faults, and invalid memory access. Program interrupts are usually handled by the operating system to prevent the program from crashing.
2. **Timer Interrupts:** Timer interrupts are generated by a timer hardware device at regular intervals to perform scheduled tasks or switch between processes. Timer interrupts are often used for scheduling processes, updating system clocks, or controlling system sleep modes.
3. **I/O Interrupts:** I/O interrupts are generated by hardware devices to signal the CPU that an I/O operation has completed. For example, an I/O interrupt may be generated by a hard drive when it has finished reading or writing data. I/O interrupts allow the CPU to continue executing other tasks while waiting for the I/O operation to complete.
4. **Hardware Interrupts:** Hardware interrupts are generated by hardware devices to request attention from the CPU. Hardware interrupts can be triggered by various events, such as key presses, mouse movements, or network packets. Handling hardware interrupts is a critical part of the operating system's job, as they allow the computer to respond to external events in real-time.

In summary, program, timer, I/O, and hardware interrupts are the main types of interrupts that can occur in a computer system. Each type of interrupt serves a different purpose and can be handled differently by the operating system. Handling interrupts efficiently is crucial for maintaining system performance and responsiveness.

Not sure:

# Addition of Interrupt cycle to Instruction cycle:

* Interrupt cycle is added to the instruction cycle to check for interrupts.
* If there is no interrupt, the processor proceeds to fetch the next instruction of the current program.
* However, if an interrupt is pending, the processor saves the context of the current program.
* The program counter is set to the starting address of the Interrupt handler routine, and the interrupt is serviced.
* After servicing the interrupt, the current program resumes, and the program counter is set to the next instruction.
* The Interrupt handler routine is a special program that is designed to handle the interrupt, and it is executed in a special mode called the interrupt mode.
* The ability to handle interrupts efficiently is essential for the CPU to perform time-critical tasks and to handle multiple tasks simultaneously.
* Interrupt priority scheme is used to determine the order in which interrupts are handled.
* Once the interrupt has been handled, the CPU resumes the normal execution of the program from the point where it was interrupted.
* In summary, the addition of the interrupt cycle to the instruction cycle allows the CPU to handle interrupts efficiently, which is necessary for the CPU to perform time-critical tasks and to handle multiple tasks simultaneously.

**Disabled Vs Enabled Interrupts:**

Interrupts can be temporarily disabled or enabled in order to manage multiple interrupts. Here are the two methods to handle multiple interrupts:

Disabled Interrupts: One method is to disable interrupts temporarily while an interrupt is being serviced. This is achieved by setting a flag called the interrupt disable flag or interrupt mask. Once an interrupt is serviced, the flag is reset, and the processor can respond to new interrupts. This method ensures that a higher-priority interrupt does not preempt a lower-priority interrupt, which could result in an incomplete or incorrect operation. However, it can lead to increased latency if the CPU spends too much time with interrupts disabled.

Enabled Interrupts: Another method is to allow interrupts to occur at any time, including during interrupt service routines. This is known as nested interrupts. The CPU keeps track of the priority of each interrupt and interrupts the current service routine if a higher-priority interrupt occurs. Nested interrupts can result in faster response times, but they can also lead to more complex software design and may be more difficult to debug.

In summary, the two methods to handle multiple interrupts are disabling interrupts and enabling interrupts. Disabling interrupts ensures that a higher-priority interrupt does not interrupt a lower-priority interrupt but can result in increased latency. Enabling interrupts allows nested interrupts and faster response times, but can also result in more complex software design and debugging. The choice of method depends on the specific requirements of the system and the characteristics of the interrupts.

# THE INTERCONNECTION STRUCTURE:

The interconnection structure in a computer system refers to the way in which the various components of the system are connected together. This includes the connection between the central processing unit (CPU), memory, input/output (I/O) devices, and other peripheral devices.

# BUS INTERCONNECTION

A bus is a communication pathway connecting two or more devices. A significant feature of a bus is that it is a shared transmission medium. Multiple devices can connect to the bus, and any signal transmitted by a device is accessible to all other devices attached to the bus. However, if two devices send signals simultaneously, their signals overlap and become garbled, leading to only one device transmitting at a time.

## Bus Design

A bus usually consists of many communication pathways, or lines, capable of transmitting signals representing binary 1 and binary 0. Each line is assigned a specific function, and several lines can be used to transmit binary digits simultaneously (in parallel). A system bus, which connects major computer components like the processor, memory, and I/O, typically consists of around 50 to hundreds of separate lines, classified into three functional groups: data, address, and control lines. The number of lines in the data bus determines how many bits can be transferred simultaneously, which plays a key factor in determining overall system performance.

## Functional Groups of a Bus

The **data lines**, collectively known as the **data bus**, provide a path for moving data among system modules. Each line can only carry one bit at a time, and the width of the data bus determines how many bits can be transferred simultaneously The address lines specify where the data on the data bus is coming from or going to. The width of the address bus determines the maximum possible memory capacity of the system, and the higher-order bits are used to select a particular module on the bus. The lower-order bits select a memory location or I/O port within the selected module. The control lines are used to control access to and the use of the data and address lines. Control signals transmit both command and timing information among system modules.

# Bus Interconnection Scheme

## Operation of the Bus

The bus is used to transfer data between different components or modules in a computer system. When a module wants to send data to another module, it must obtain use of the bus. The bus grants access to the module and allows it to transfer the data to the destination module. If a module wants to request data from another module, it must obtain use of the bus, request the data, and wait for a response.

## Types of Buses

Modern computers tend to have all major components on board, with more elements on the same chip as the processor. There are two types of buses used in a computer system: on-chip bus and on-board bus.

**On-Chip Bus:** The on-chip bus is used to connect the processor and cache memory. Since the cache memory is located on the same chip as the processor, the on-chip bus provides a faster data transfer rate between the processor and cache memory.

**On-Board Bus:** The on-board bus is used to connect the processor to main memory and other components. It is used to transfer data between different components on the motherboard, such as the video card, sound card, and network card. The on-board bus provides a slower data transfer rate compared to the on-chip bus.