Table of Contents

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.

3.1 Computer Components

Contemporary computers use the von Neumann architecture, which was developed by John von Neumann at Princeton. The design is based on three main concepts.

Firstly, data and instructions are stored in a single memory that can be read and written to.

Secondly, the contents of the memory can be addressed by location, without considering the type of data stored.

Finally, instructions are executed one after the other in a sequential manner, unless explicitly modified

Hard Wired Program

Hardware components can be connected together to perform specific computations like comparing, adding, or multiplying numbers. Connecting various hardware components to perform a specific computation is like programming. The resulting program is a hardwired program, meaning the hardware is customized to perform a specific function. Each different function requires a different hardware structure.

General Purpose Configuration of Hardware

In a hardwired program system, data is accepted and output is produced. To create a more general-purpose configuration of arithmetic and logic functions, we can construct a hardware system that performs various functions on data based on control signals applied to it. This hardware system can accept data like a hardwired system, but it can also accept control signals, unlike hardwired programs. This means that instead of rewiring the system for different functions, we can apply a new set of control signals to perform different operations.

How to apply Control Signals

To execute a program, a sequence of steps is needed, where each step involves performing some arithmetic or logic function. For each step, a unique set of control signals is required, which may be the same as previous steps. To provide these control signals, we assign a unique code for each set of signals and add a segment into the general purpose hardware configuration. When a code is accepted, it interprets the code and then generates the corresponding set of control signals required for that step.

This new method of programming, which involves using a sequence of codes or instructions, is called software.

Components cont….

To make a computer function properly, several additional components are required with CPU. These components include an input module, which takes data and converts it into a format that the computer can understand. An output module is also necessary to convert the signals generated by the computer into a format that a user can understand.

Additionally, a memory or main memory is necessary to temporarily hold instructions and data. This helps the computer to access information quickly and efficiently. All these components work together to ensure that the computer can run programs properly and perform the tasks it was designed for.

Communication of CPU with Memory

The CPU communicates with memory by exchanging data through two registers: the MAR, which holds the address of the current read/write location in memory, and the MBR, which holds the data to be written into or accessed from memory.

These registers allow the CPU to exchange data with the memory efficiently, ensuring that the computer system can perform its tasks quickly and accurately.

Communication of CPU with I/O

- CPU communicates with I/O devices through an I/O module
- The I/O module uses two registers to communicate with the CPU: the I/OAR (I/O Address Register) and the I/OBR (I/O Buffer Register)
- The I/OAR specifies a particular I/O device that the CPU wants to communicate with
- The I/OBR is used to exchange data between the I/O module and the CPU
- I/O buffers are also present to hold the data temporarily.

3.2 COMPUTER FUNCTION

A computer's main function is to execute a program, which is a set of instructions stored in its memory. The processor reads each instruction and performs the necessary operations, using a process called the instruction cycle. This cycle consists of two parts: the fetch cycle, where the next instruction is fetched from memory, and the execute cycle, where the instruction is carried out. This process repeats until the program is finished or an error occurs. The computer will stop running the program if it's turned off, there's an unfixable error, or it comes across an instruction that tells it to stop.

Fetch Cycle

During the fetch cycle, the processor fetches an instruction from memory. The address of the instruction to be fetched next is stored in the program counter (PC). The processor increments the PC after each instruction fetch, so it can get the next instruction in sequence. The fetched instruction is then loaded into the instruction register (IR). Finally, the processor interprets the instruction and performs the necessary action. This process is repeated for each instruction in the program until the program is finished or an error occurs.

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.

WITHOUT 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.

WITH 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 occur when an external device, such as a keyboard or mouse, sends data to the processor.

4. **Hardware Interrupts:** Hardware interrupts occur when there is a problem with a hardware device, such as a disk drive or printer. Hardware interrupts can also be caused by power failures, overheating, or other hardware-related issues.

Interrupts and the Instruction Cycle

A processor is engaged while I/O is in progress, but when the device is ready to accept more data from the processor, an I/O module sends an interrupt signal. The current state of the program is saved and control is transferred to an interrupt handler. After servicing the interrupt, the program resumes from where it left off. This process is called the interrupt cycle.

Addition of Interrupt cycle to Instruction cycle:

The interrupt cycle is an important part of the computer's operation. It is added to instruction cycle to check for any interrupts that may occur while the processor is running a program. If there is no interrupt, the CPU proceeds to fetch the next instruction of the current program. But if there is a pending interrupt, the processor saves the current program's state and sets the program counter to the starting address of the interrupt handler routine. Then, the interrupt is serviced. After servicing the interrupt, the original/current program resumes where it left, and the program counter is set to the next instruction.

This ensures that important events, such as user input or hardware signals, can be handled promptly without disrupting the normal flow of the program.

Disabled Vs Enabled Interrupts – Multiple 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.

Bus arbitration

Bus arbitration is a process that allows multiple modules to control a bus. For example, a CPU and DMA controller may both want to control the bus at the same time. However, only one module can control the bus at any given time. Bus arbitration is the process of determining which device gets to control the bus at

any given time. There are two types of bus arbitration: centralized and distributed. Centralized arbitration involves a single hardware device controlling bus access, while distributed arbitration involves every device taking part in choosing the new bus master

Timing Synchronous-Asynchronous

Timing refers to how events are coordinated on the bus, which can be either synchronous or asynchronous. In synchronous timing, events are determined by clock signals, and the control bus includes a clock line. On the other hand, in asynchronous timing, the occurrence of one event on the bus follows and depends on the occurrence of a previous event, without relying on a central clock signal.