NEERAJ®

MCS-202
Computer Organisation

Chapter Wise Reference Book
Including Many Solved Sample Papers

Based on

I.G.N.O.U.

& Various Central, State & Other Open Universities

By: *Anand Prakash Srivastava*

NEERAJ PUBLICATIONS
(Publishers of Educational Books)

Mob.: 8510009872, 8510009878 E-mail: info@neerajbooks.com
Website: www.neerajbooks.com

MRP ₹ 300/-

Content

COMPUTER ORGANISATION

S.No.	*Chapterwise Reference Book*	*Page*

■■

Sample Preview of the Solved Sample Question Papers

Published by:

NEERAJ PUBLICATIONS

www.neerajbooks.com

QUESTION PAPER

June – 2024

(Solved)

COMPUTER ORGANISATION

MCS-202

Time: 3 Hours] ***[Maximum Marks : 100***
Weightage : 70%

Note: *Question* ***No. 1*** *is compulsory. Attempt any* ***three*** *questions from the rest.*

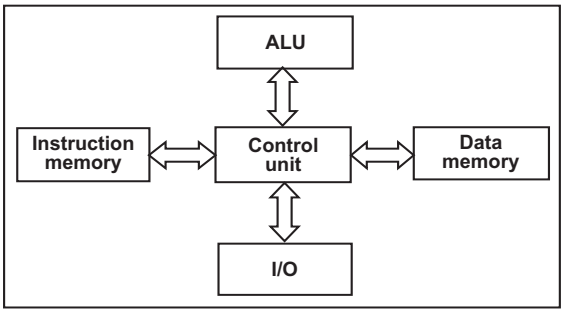
Q. 1. ***(a)*** **Explain Harvard architecture with the help of a diagram.**

Ans. Ref.: See Chapter-1, Page No. 4, 'Harvard Architecture'.

Also Add: Harvard architecture is a type of computer architecture that separates its memory into two parts so data and instructions are stored separately. The architecture also has separate buses for data transfers and instruction fetches. This allows the CPU to fetch data and instructions at the same time. Today, processors using Harvard architecture use a modified form so they can achieve a greater performance. Some modified forms allow the support of tasks like loading a program from secondary storage (opposed to RAM) as data then executing it. In some systems, instructions are stored in read-only memory and data in read-write memory. This architecture is sometimes used within the CPU to handle its caches, but it is less used with main memory because of complexity and cost. It is used extensively with embedded Digital Signal Processing (DSP) systems. DSP systems include audio and speech signal processing, sonar and radar signal processing machines, biomedical signal processing, seismic data processing and digital image processing.

The key feature:

1. The two different memories can have different characteristics: for example, in embedded systems, instructions may be held in read-only memory while data may require read-write memory.
2. In some systems, there is much more instruction memory than data memory, so a larger word size is used for instructions.
3. The instruction address bus may be wider than the data bus.



(b) **Perform the following conversions:**

(i) **Decimal (873)10 to Binary**

Ans. Step: Divide by 2 repeatedly and write the remainders in reverse.

873 ÷ 2 = 436 remainder 1
436 ÷ 2 = 218 remainder 0
218 ÷ 2 = 109 remainder 0
109 ÷ 2 = 54 remainder 1
54 ÷ 2 = 27 remainder 0
27 ÷ 2 = 13 remainder 1
13 ÷ 2 = 6 remainder 1
6 ÷ 2 = 3 remainder 0
3 ÷ 2 = 1 remainder 1
1 ÷ 2 = 0 remainder 1

$(873)_{10} = (1101101001)_2$

(ii) **Decimal (384)10 to Hexadecimal**

Ans. Step: Divide by 16 repeatedly and write remainders in reverse.

384 ÷ 16 = 24 remainder 0
24 ÷ 16 = 1 remainder 8
1 ÷ 16 = 0 remainder 1

$(384)_{10} = (180)_{16}$

(iii) **Hexadecimal $(FAB)_{16}$ to Octal**

Ans. Step 1: Convert Hex → Binary

F = 1111
A = 1010
B = 1011

So, $(FAB)_{16} = 111110101011_2$

Step 2: Group into 3 bits from right → Convert to Octal

111 110 101 011
= 7 6 5 3

$(FAB)_{16} = (7653)_8$

(iv) **ASCII string 'the escape sequence' to UTF8**

Ans. In UTF-8, basic ASCII characters (U+0000 to U+007F) are encoded identically, i.e., each character uses 1 byte and maps directly.

So, for the string "the escape sequence", just write UTF-8 hex codes of each character:

Characters:
t h e e s c a p e s e q u e n c e

ASCII / UTF-8 (Hex):

74 68 65 20 65 73 63 61 70 65 20 73 65 71 75 65 6E 63 65

UTF-8 Encoding (Hex):

74 68 65 20 65 73 63 61 70 65 20 73 65 71 75 65 6E 63 65

(v) Octal $(765)_8$ to Decimal

Ans. Step: Expand using powers of 8

$= 7\times8^2 + 6\times8^1 + 5\times8^0$

$= 7\times64 + 6\times8 + 5$

$= 448 + 48 + 5$

$= 501$

$(765)_8 = (501)_{10}$

***(c)* Differentiate between CLV and CAV type of disk organizations.**

Ans. Ref.: See Chapter-5, Page No. 54, Q. No. 9.

***(d)* Explain two-way set associative Cache mapping with a suitable example.**

Ans. Two-way Set Associative Cache Mapping is a type of cache memory organization that lies between Direct Mapping and Fully Associative Mapping. It combines the simplicity of direct mapping with some flexibility of fully associative mapping.

In Two-Way Set Associative Mapping, the cache is divided into sets, and each set contains two cache lines (blocks). A block from main memory can be placed in any of the two lines in a specific set.

Also Add: Ref.: See Chapter-6, Page No. 60, Q. No. 6.

***(e)* Differentiate between RISC processor and CISC processor. Give utility of each.**

Ans. Ref.: See Chapter-12, Page No. 110, 'Introduction to Risc', 'Compared to Complex Instruction Set Computers (CISC)' and 'RISC processors'.

***(f)* What is an interrupt ? Explain the use of interrupt in input/output with the help of an example.**

Ans. Ref: See Chapter-7, Page No. 70, Q. No. 5.

Also Add:

Interrupt driven I/O is an alternative scheme dealing with I/O. Interrupt I/O is a way of controlling input/output activity whereby a peripheral or terminal that needs to make or receive a data transfer sends a signal. This will cause a program interrupt to be set. At a time appropriate to the priority level of the I/O interrupt. Relative to the total interrupt system, the processors enter an interrupt service routine. The function of the routine will depend upon the system of interrupt levels and priorities that is implemented in the processor. The interrupt technique requires more complex hardware and software, but makes far more efficient use of the computer's time and capacities. Figure 2 shows the simple interrupt processing.

For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports or memory mapping.

For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer.

Here the CPU works on its given tasks continuously. When an input is available, such as when someone types a key on the keyboard, then the CPU is interrupted from its work to take care of the input data. The CPU can work continuously on a task without checking the input devices, allowing the devices themselves to interrupt it as necessary.

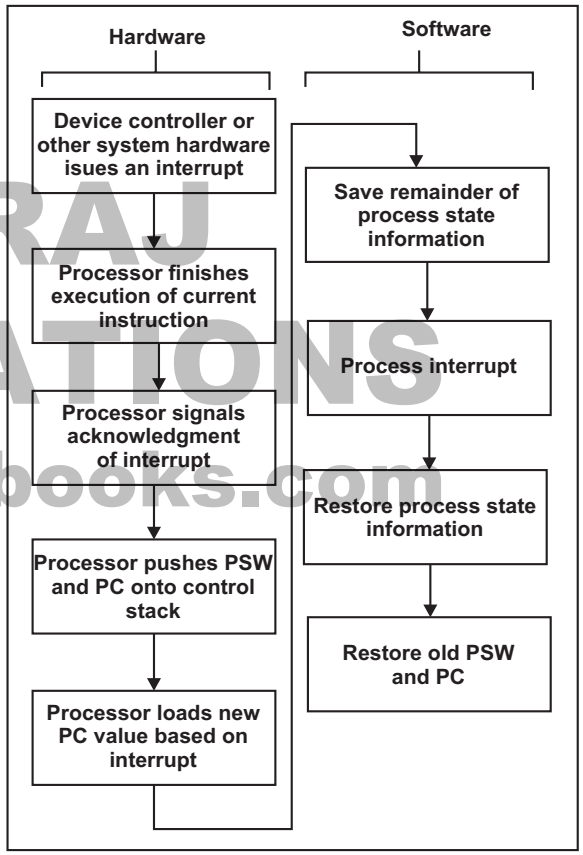


Fig.: Simple Interrupt Processing

***(g)* List the various register categories involved with 8086 microprocessor. Also, give the role of each type of register category.**

Ans. Ref.: See Chapter-13, Page No. 116, 'Structure of 8086 CPU'.

***(h)* Compare direct addressing mode and indirect addressing mode. Give example for each.**

Ans. Ref: See Chapter-13, Page No. 117, 'Addressing Modes'.

Sample Preview of The Chapter

Published by:

NEERAJ PUBLICATIONS

www.neerajbooks.com

COMPUTER ORGANISATION

Computer System

1

INTRODUCTION

In today's competitive world, businesses rely heavily on advanced information technology to streamline their processes and remain competitive. Computers have become an integral part of our daily lives, enabling us to access health services, banking, education, and government services efficiently. This chapter introduces the fundamental concepts of computer systems, including their structure, history, and various architectures. It also explains how instructions are executed in a computer, which is a crucial aspect of understanding how computers function.

The chapter begins with a brief history of computers, followed by an exploration of the basic structure of a computer system, including the CPU, memory, and input/output devices. It then delves into the execution of instructions and the instruction cycle, which are essential for understanding how a computer processes data. Finally, the chapter discusses various computer architectures, such as von Neumann, Harvard, RISC, and multicore architectures, as well as the emerging field of mobile architecture.

CHAPTER AT A GLANCE

A BRIEF HISTORY

Subject	1st Generation	2nd Generation	3rd Generation	4th Generation	5th Generation
Period	1940-1956	1956-1963	1964-1971	1971-present	present & beyond
Circuitry	Vacuum tube	Transistor	Integrated Chips (IC)	Microprocessor (VLSI)	ULSI (Ultra Large Scale Integration) Technology
Memory Capacity	20 KB	128KB	1MB	Magnetic Core Memory, LSI and VLSI. High Capacity	ULSI
Processing Speed	300 IPS instructions Per sec.	300 IPS	1MIPS (1 million inst. Per sec.)	Faster than 3rd generation	Very fast
Programming Language	Machine. Language	Assembly language & early high-level languages (FORTRAN, COBOL, ALGOL)	C.C++	Higher level languages C.C++, Java	All the Higher level languages..Neural networks.
Example of Computers	UNIVAC, EDVAC	IBM 1401, IBM 7094, CDC 3600,D UNIVAC 1108	IBM 360 series, 1900 series	Pentium series, Multimedia,	Artificial Intelligence, Robotics

Semi-conductor Chips and Computers

A computer system is built using semiconductor materials, primarily transistors, which enable large-scale circuit integration on a single chip. Over the past five decades, advancements in integration technology have led to exponential improvements in processing power and memory capacity.

Initially, Small-scale Integration (SSI) featured only a few transistors per chip. With technological progress, chips evolved through Medium-Scale (MSI), Large-Scale (LSI), Very Large-Scale (VLSI), and now ultra Large-scale Integration (ULSI), significantly increasing transistor density.

STRUCTURE OF A COMPUTER

A computer system consists of several key components that work together to process data and execute instructions. These components include the **Central Processing Unit (CPU)**, **memory**, **input/output devices**, and **datapaths**.

The CPU

The CPU is the brain of the computer, responsible for executing instructions stored in the computer's memory. It consists of three main components:

- **Registers:** Temporary storage locations within the CPU that hold data, addresses, and instructions.
- **Arithmetic Logic Unit (ALU):** Performs arithmetic and logical operations on data.
- **Control Unit (CU):** Manages the execution of instructions by controlling the flow of data between the CPU, memory, and I/O devices.

The CPU fetches instructions from memory, decodes them, executes the operations, and stores the results back in memory or registers.

Register Set

Registers are high-speed storage locations within the CPU used to store temporary data, addresses, and instructions. There are different types of registers:

- **Program Counter (PC):** Holds the address of the next instruction to be executed.
- **Instruction Register (IR):** Stores the instruction currently being executed.
- **General-Purpose Registers:** Used for storing data or addresses during computations.
- **Status Registers:** Store information about the current state of the CPU, such as flags indicating the result of an operation.

Datapath

The datapath is the pathway through which data flows between the CPU, memory, and I/O devices. It consists of internal and external datapaths:

- **Internal Datapaths:** Transfer data between registers and the ALU within the CPU.
- **External Datapaths:** Transfer data between the CPU and memory or I/O devices, typically using a system bus.

Control Unit

The Control Unit (CU) is responsible for managing the execution of instructions. It generates control signals that coordinate the activities of the CPU, memory, and I/O devices. The CU ensures that instructions are fetched, decoded, and executed in the correct sequence.

Memory Unit and I/O Devices

The memory unit stores data and instructions that the CPU needs to execute programs. Memory is organized in bytes, with larger units such as kilobytes (KB), megabytes (MB), and gigabytes (GB) used to measure its capacity. **Random Access Memory (RAM)** is the primary memory used by the CPU to store data and instructions temporarily.

Input/output (I/O) devices allow the computer to interact with the external world. Input devices, such as keyboards and mice, provide data to the computer, while output devices, such as monitors and printers, display or produce the results of computations.

What is an Instruction?

An instruction is a binary code that tells the CPU what operation to perform. Instructions typically consist of an **opcode** (operation code) that specifies the operation (e.g., addition, subtraction) and **operands** that specify the data or addresses on which the operation is to be performed. Instructions are stored in memory and fetched by the CPU for execution.

HOW ARE INSTRUCTIONS EXECUTED?

The execution of instructions involves several steps:

1. Fetching: The CPU fetches the instruction from memory using the address stored in the Program Counter (PC).

2. Decoding: The Control Unit decodes the instruction to determine the operation to be performed.

3. Reading Operands: The CPU reads the operands (data) from memory or registers.

4. Executing: The ALU performs the operation specified by the instruction.

5. Storing Results: The result of the operation is stored in a register or memory location.

This process is repeated for each instruction in a program.

INSTRUCTION CYCLE

A program consists of instructions stored in memory, executed sequentially by the processor. The execution follows *four* phases:

1. **Fetching**
2. **Decoding**
3. **Reading Operands**
4. **Execution**

Fetching the Instruction: Instructions are stored in RAM, and the Program Counter (PC) holds their location. The instruction is fetched and placed into the Instruction Register.

Decode the Instruction: The CPU interprets the operation code and determines operand locations in memory.

Read the Operands from Memory: Once operand addresses are identified, they are loaded into CPU registers for processing.

Execute the Instruction: The CPU performs the required operation, and results are stored in temporary locations.

Interrupts

An interrupt temporarily halts program execution after completing the current instruction. It improves efficiency by allowing the CPU to respond to events.

Steps to process an interrupt:

- Identify the interrupt source.
- Execute the Interrupt Service Routine (ISR).
- Hold the current program.
- Resume execution after ISR completion.

Interrupts and Instruction Cycle

- CPU executes program "X" and is decoding the i^{th} instruction.
- An interrupt occurs during this time.
- CPU completes the i^{th} instruction, saves register values and PC.
- It identifies and executes the Interrupt Service Routine (ISR).
- After ISR completion, CPU restores registers and PC, resuming execution from $(i+1)^{th}$ instruction.

VARIOUS COMPUTER ARCHITECTURE

Computer architecture refers to the design and organization of a computer system. Different architectures have been developed to optimize performance, efficiency and cost.

Von-Neumann Architecture

The **von-Neumann architecture**, developed by John von Neumann, is the foundation of most modern computers. It uses a single memory space to store both instructions and data. The CPU fetches instructions and data from the same memory, which can create a bottleneck known as the **von-Neumann bottleneck**. Despite this limitation, the von-Neumann architecture is simple and widely used.

Harvard Architecture

The **Harvard Architecture** uses separate memory spaces for instructions and data, allowing the CPU to fetch both simultaneously. This architecture is more complex but offers improved performance, especially in applications requiring high-speed data processing, such as digital signal processing.

Instruction Set Architecture (ISA)

The **Instruction Set Architecture (ISA)** defines the set of instructions that a CPU can execute. It specifies the types of operations, instruction formats, and data types supported by the processor. Examples of ISAs include Intel's x86, ARM, and MIPS.

RISC

RISC (Reduced Instruction Set Computing) is a design philosophy that simplifies the instruction set to improve performance. RISC processors execute simple instructions in a single clock cycle, reducing hardware complexity and increasing speed. In contrast, **CISC (Complex Instruction Set Computing)** processors use complex instructions that require multiple clock cycles.

Multiprocessor And Multicore Architecture

Multiprocessor systems use multiple CPUs to improve performance by executing tasks in parallel. **Multicore processors** take this concept further by integrating multiple CPU cores on a single chip. Each core can execute instructions independently, allowing for greater parallelism and efficiency.

Mobile Architecture

Mobile architecture refers to the design of processors and systems used in mobile devices, such

as smartphones and tablets. These devices typically have two processing units: a **Communications Processing Unit** for handling calls and network operations, and an **Applications Processing Unit** for running apps. Mobile architectures are optimized for power efficiency and performance in compact devices.

CHECK YOUR PROGRESS

Q. 1. State True or False.

***(a)* Byte consists of 8 bits, and it may represent a character.**

Ans. True.

***(b)* A character representation requires at least one byte.**

Ans. True.

***(c)* One bit is an independent unit and can be accessed independently.**

Ans. False.

***(d)* A machine can have two paths called internal and external.**

Ans. True.

Q. 2. Explain the concept of an Instruction.

Ans. An **instruction** in computing is a fundamental command or operation that a Computer's Processor (CPU) understands and executes. Instructions are part of a program and are written in machine language or assembly language, which the CPU interprets to perform tasks. Each instruction directs the CPU to perform a specific operation, such as arithmetic calculations, data transfer, logical comparisons, or control flow changes.

Structure of an Instruction

An instruction typically consists of two main parts:

Opcode (Operation Code): This specifies the action to be performed, such as ADD (addition), SUB (subtraction), MOV (move data), or JMP (jump to another instruction).

Operands: These specify the data or memory addresses involved in the operation. The operands can be:

- Immediate values (constants)
- Memory addresses (locations where data is stored)
- CPU registers (fast storage locations within the CPU)

Q. 3. Why does a computer need CU, ALU, memory and I/O devices?

Ans. A computer requires the **Control Unit (CU), Arithmetic Logic Unit (ALU), memory, and Input/Output (I/O) devices** because each of these components plays a crucial role in ensuring the smooth and efficient operation of the system.

The **Control Unit (CU)** is responsible for managing and coordinating all activities within the computer. It acts as the brain of the CPU, directing how data moves between different components and ensuring that instructions are executed in the correct sequence. It fetches instructions from memory, decodes them, and then signals the appropriate components to execute the required operations. Without the CU, the computer would not be able to interpret and execute software programs.

The **Arithmetic Logic Unit (ALU)** is the component that performs all mathematical calculations and logical operations. It handles basic arithmetic operations such as addition, subtraction, multiplication, and division, as well as logical comparisons like AND, OR, and NOT operations. The ALU enables decision-making within the computer by evaluating conditions and determining the appropriate actions based on logical statements. This is fundamental for processing data, running applications, and executing complex computations.

Memory is essential for temporarily and permanently storing data and instructions. The computer primarily uses **RAM (Random Access Memory)** to hold active programs and data that are being processed. RAM provides fast access to data, ensuring that the CPU can retrieve and execute instructions quickly. Additionally, the computer relies on **secondary storage** devices, such as hard drives and Solid-state Drives (SSDs), to store data permanently. Without memory, the computer would have to retrieve every instruction and piece of data from slow external storage, making processing extremely inefficient.

Input and Output (I/O) devices serve as the interface between the user and the computer. **Input devices**, such as keyboards, mice, scanners, and microphones, allow users to enter commands, text, and data into the system. **Output devices**, such as monitors, printers, and speakers, display or convey the results of processing to the user. These devices enable communication between the user and the computer, making interaction possible.