Answer Starts

From Page 8

No. of Printed Pages : 6 **MCS–202**

POST GRADUATE DIPLOMA IN

COMPUTER APPLICATIONS

(PGDCA–NEW)

Term-End Examination

June, 2025

MCS-202 : COMPUTER ORGANIZATION

Time : 3 Hours *Maximum Marks : 100*

Weightage : 70%

Note : *Question No.* ***1*** *is compulsory and carries 40 marks. Attempt any* ***three*** *questions from Q. No.* ***2*** *to Q. No.* ***5****.*

---

1. (a) Write the binary equivalent for the following decimal numbers : 6

(i) –23456

(ii) 299

(iii) 17.89

(b) Explain the importance of locality of reference with the help of an example. 5

(c) What is the role of a 'flag' register in 8086 microprocessor ? Can it be used as a general purpose register ? 5

(d) List and explain any *four* data transfer instructions of a computer. 4

(e) Add the following decimal numbers by converting them to 8 bit signed 2's complement notation : 4

(i) + 56 and – 56

(ii) + 121 and + 8

(f) What is an I/O interface ? List major functions of I/O interface. 4

(g) With the help of a block diagram, explain the concept of Direct Memory Access (DMA). 4

(h) Write an 8086 assembly program that interchanges the values of two memory locations. 4

(i) How are characters recognised by voice-based input devices ? 4

2. (a) Simplify the function : 5

f [illegible] A [illegible] + B [illegible] A.B [illegible]

(b) Define SOP and POS form of Boolean functions with the help of an example of each. 5

(c) What is a flip-flop ? How is it different from a latch ? Also, explain the working

MCS 202- June 2025

of the following flip-flops along with necessary diagrams : 10

(i) Master-Slave flip-flop

(ii) Edge-Triggered flip-flop

3. (a) Assume that a computer system has the following memories : 6

(i) RAM 64 words with 16-bits word.

(ii) Cache of 8 blocks (block size – 32 bits).

Find in which location of Cache memory a decimal address ‘21’ can be found if direct mapping is used.

(b) List the advantages and disadvantages of Associative Memory. 4

(c) Explain how RAID technology enhances the performance and reliability of data storage. Also, summarize the characteristics of any *three* RAID-levels. 10

4. (a) Explain the role of the following types of registers : 6

(i) General purpose register

(ii) Address register

(iii) Data register

(b) With the help of a block diagram, explain the fixed-point arithmetic logic unit. 8

(c) Explain the process of execution of a micro-program. 6

MCS 202- June 2025

4×5=20

5. Write short notes on the following :

(a) Indirect Addressing Mode

(b) COM Program

(c) Dynamic Random Access Memory (DRAM)

(d) Von Neumann Architecture

× ××× ×

1. (a) Write the binary equivalent for the following decimal numbers :

(i) –23456

The binary equivalent of −23456 can be obtained by first converting 23456 into binary and then taking its 2’s complement.

23456 in binary = 0101 1011 1010 0000 (16-bit representation)

Now take 2’s complement to represent the negative number:

1’s complement: 1010 0100 0101 1111
Add 1: 1010 0100 0110 0000

So, the binary equivalent of −23456 (in 16-bit 2’s complement form) is:

1010010001100000

(ii) 299

To convert 299 into binary, divide the number repeatedly by 2 and record the remainders.

299 ÷ 2 = 149 remainder 1
149 ÷ 2 = 74 remainder 1
74 ÷ 2 = 37 remainder 0
37 ÷ 2 = 18 remainder 1
18 ÷ 2 = 9 remainder 0
9 ÷ 2 = 4 remainder 1
4 ÷ 2 = 2 remainder 0
2 ÷ 2 = 1 remainder 0
1 ÷ 2 = 0 remainder 1

Now writing the remainders from bottom to top:

$\mathbf{299_{10} = 100101011_2}$

(iii) 17.89

To convert 17.89 into binary, convert the integer part and fractional part separately.

Integer part:

17 in binary = 10001

Fractional part:

$0.89 \times 2 = 1.78 \rightarrow 1$

$0.78 \times 2 = 1.56 \rightarrow 1$

$0.56 \times 2 = 1.12 \rightarrow 1$

$0.12 \times 2 = 0.24 \rightarrow 0$

$0.24 \times 2 = 0.48 \rightarrow 0$

$0.48 \times 2 = 0.96 \rightarrow 0$

$0.96 \times 2 = 1.92 \rightarrow 1$

$0.92 \times 2 = 1.84 \rightarrow 1$

So fractional binary ≈ .11100011 (approximate)

Final answer:

$\mathbf{17.89_{10} \approx 10001.11100011_2}$

(b) Explain the importance of locality of reference with the help of an example.

Locality of reference is a property of programs in which the same memory locations are accessed repeatedly within a short period of time. It plays an important role in improving the performance of a computer system, especially in cache memory design.

There are two main types of locality of reference. Temporal locality means that if a particular memory location is accessed once, it is likely to be accessed again soon. Spatial locality means that if a memory location is accessed, nearby memory locations are also likely to be accessed.

For example, consider a loop that processes elements of an array. When the loop runs, it repeatedly accesses consecutive memory locations of the array. This shows spatial locality. Also, the loop instructions themselves are executed again and again, which shows temporal locality.

Because of locality of reference, cache memory can store frequently used data and instructions, reducing access time and improving overall system performance.

(c) What is the role of a 'flag' register in 8086 microprocessor ? Can it be used as a general purpose register ?

The flag register in the 8086 microprocessor is a special purpose register that indicates the status of the processor and the results of arithmetic and logical operations. It is a 16-bit register in which each bit represents a specific condition or control feature.

The flag register is divided into two types of flags. Status flags show the result of an operation. These include Carry Flag (CF), Zero Flag (ZF), Sign Flag (SF), Overflow Flag (OF), Auxiliary Carry Flag (AF), and Parity Flag (PF). Control flags are used to control processor operations. These include Trap Flag (TF), Interrupt Flag (IF), and Direction Flag (DF).

The role of the flag register is to help the processor make [illegible] such as check[illegible] carry occurred, [illegible]

The flag register cannot be used as a general purpose register because it is designed only to store status and control information. It does not store general data or operands like AX, BX, CX, or DX registers.

(d) List and explain any four data transfer instructions of a computer.

Data transfer instructions are those instructions that are used to move data from one location to another inside a computer system. These instructions do not change the actual data. They only copy the data between registers, memory, or input and output devices. They play a very important role in program execution because almost every operation requires data movement before processing.

Four commonly used data transfer instructions are explained below.

MOV instruction is the most widely used data transfer instruction. It is used to copy data from a source location to a destination location. The source can be a register, memory location, or immediate data, and the destination can be a register or memory. For example, MOV AX, BX copies the contents of register BX into register AX. The original value in BX remains unchanged.

LOAD instruction is used to transfer data from memory into a register. It is helpful when the processor needs to fetch data stored in memory for processing. For example, LOAD R1, 2000H means the data stored at memory address 2000H is transferred into register R1.

STORE instruction performs the opposite function of LOAD. It transfers data from a register into a memory location. This is used when processed results need to be saved back into memory. For example, STORE R2, 3000H stores the contents of register R2 into memory location 3000H.

XCHG instruction is used to exchange the contents of two registers or a register and a memory location. It is useful when two values need to be swapped without using an extra register. For example, XCHG AX, BX exchanges the values stored in AX and BX registers.

Thus, data transfer instructions are essential because they enable smooth movement of data between different parts of the computer system, which is necessary for proper execution of programs.

(e) Add the following decimal numbers by converting them to 8 bit signed 2's complement notation :

To add the given decimal numbers using 8-bit signed 2's complement notation, we first convert each decimal number into its 8-bit 2's complement binary form and then perform [illegible] a[illegible]tio[illegible]

(i) +5[illegible] and [illegible]5[illegible]

First convert +56 into binary:

56_{10} = 00111000 (8-bit binary)

Now c[illegible]

+56 = 00111000

1's complement = 11000111
Add 1 = 11001000

So, −56 = 11001000

Now perform binary addition:

00111000

- 11001000
- = 00000000 (carry ignored in 8-bit addition)

Final result = 00000000 which represents 0 in decimal

(ii) +121 and +8

Convert both numbers into 8-bit binary:

121_{10} = 01111001
8_{10} = 00001000

Now add:

01111001

- 00001000
- = 10000001

Here the result is 10000001, which indicates overflow because both numbers were positive but the result has a sign bit of 1 (negative). Therefore, overflow occurs in this case.

Thus, the results are:

(i) Sum = 00000000 (no overflow)
(ii) Sum = 10000001 (overflow occurs)

(f) What is an I/O interface ? List major functions of I/O interface.

An I/O interface is a hardware component that acts as a communication link between the central processing unit (CPU) and input or output devices. Since the CPU and peripheral devices operate at different speeds and use different data formats, an I/O interface is required to ensure proper coordination and data transfer between them.

The I/O interface works as an intermediary that translates signals, manages timing differences, and controls the flow of data between the processor and external devices such as keyboards, printers, disks, and monitors. Without an I/O interface, direct communication between the CPU and peripherals would be difficult and unreliable.

The major functions of an I/O interface are as follows. Address decoding is one of its primary functions. The interface re[illegible] the CPU and determines whether the communication request is mea[illegible].

Data buffering is another important function. Since peripheral devices are usually slower than the CPU, the interface temporarily stores data in buffers so that the speed [illegible]

Control a[illegible]ed by the interface. It generates control signals required for coordinating data transfer between the CPU and the device and ensures proper synchronization.

Data conversion is performed when necessary. The interface converts data formats, such as converting serial data into parallel form or vice versa, depending on the requirements of the CPU and the device.

Error detection is another function of the I/O interface. It checks for errors during data transfer and ensures reliable communication.

Thus, an I/O interface plays a crucial role in enabling smooth, efficient, and accurate communication between the processor and peripheral devices.

(g) With the help of a block diagram, explain the concept of Direct Memory Access (DMA).

Direct Memory Access (DMA) is a technique that allows an input or output device to transfer data directly to or from the main memory without continuous involvement of the CPU. It is used to improve the speed of data transfer and to reduce the workload of the processor.

In normal data transfer, the CPU controls every step of the communication between memory and I/O devices. This makes the CPU busy and slows down system performance. DMA solves this problem by introducing a special hardware unit called the DMA controller.

The DMA controller temporarily takes control of the system buses and performs data transfer directly between memory and the I/O device. During this time, the CPU is free to perform other tasks, which improves overall system efficiency.

The working of DMA can be explained as follows. First, the CPU initializes the DMA controller by providing the starting memory address, the number of bytes to be transferred, and the direction of transfer. Then the DMA controller requests control of the system bus from the CPU. After receiving permission, it transfers data directly between the I/O device and memory. Once the transfer is complete, it sends an interrupt signal to the CPU to indicate completion.

The block diagram of DMA consists of the CPU, main memory, I/O device, and DMA controller. The DMA controller is connected between the I/O device and the memory through the system bus. It includes an address register, data register, control logic, and a byte count register.

Thus, Direct Memory Access is an efficient method of high-speed data transfer that reduces CPU involvement and enhances overall system performance.

(h) Write an 8086 assembly program that inter... locations.

An 8086 assembly program can interchange the values of two memory locations by using a temporary register. The pr... memory locati... e into another ... back in opposite locations.

Program to interchange the values of two memory locations:


```
MOV AX, [2000H]
MOV BX, [2001H]
MOV [2000H], BX
MOV [2001H], AX
```

In this program, AX and BX are used as temporary registers. The value stored at memory location 2000H is first moved into AX. Then the value at memory location 2001H is moved into BX. After that, the contents of BX are stored into memory location 2000H, and the contents of AX are stored into memory location 2001H.

Thus, the values of the two memory locations are successfully interchanged using simple data transfer instructions in 8086 assembly language.

(i) How are characters recognised by voice-based input devices ?

Voice based input devices recognise characters through a process called speech recognition. In this process spoken words are converted into digital signals and then interpreted by the computer system to identify the corresponding characters or commands.

The recognition process begins when a user speaks into a microphone. The microphone converts the sound waves into electrical signals. These signals are then converted into digital form using an analog to digital converter so that the computer can process them.

After digitisation, the system performs feature extraction. In this step, the important characteristics of the speech signal such as frequency, pitch, and amplitude are analysed. These features help the system distinguish between different sounds and words.

Next, pattern matching is carried out. The extracted speech features are compared with stored speech patterns in a database. The system uses algorithms to find the closest match between the spoken input and the stored patterns.

Modern voice recognition systems also use language processing techniques. These techniques consider grammar, context, and probability of word occurrence to improve accuracy and correctly recognise characters and words.

Finally, once the system identifies the spoken sound, it converts it into corresponding text characters or executes the related command.

Thus, voice based input devices recognise characters through stages of sound capture, digitisation, feature extraction, pattern matching, and language processing, enabling users to interact with computers using speech.

2. (a) Simplify the function :

To simplify the Boolean function f = ((A' + B) + (A · B')')', we proceed step by step using Boolean algebra laws.

Given f

f = ((A' + B)

First simplify the term $(A \cdot B')'$:

Using De Morgan's law:

$$(A \cdot B')' = A' + (B')'$$
$$= A' + B$$

Now substitute back into the expression:

$$f = ((A' + B) + (A' + B))'$$

Since $(A' + B) + (A' + B) = A' + B$ (Idempotent law),

So the expression becomes:

$$f = (A' + B)'$$

Now apply De Morgan's law again:

$$(A' + B)' = (A')' \cdot B'$$
$$= A \cdot B'$$

Therefore, the simplified function is:

$f = A \cdot B'$

(b) Define SOP and POS form of Boolean functions with the help of an example of each.

SOP form means Sum of Products form, and POS form means Product of Sums form. These are standard ways of representing Boolean functions in digital logic design.

In the SOP form, a Boolean expression is written as a sum, which means OR operation, of two or more product terms. Each product term is formed by ANDing variables or their complements. In other words, first the variables are combined using AND operations, and then the resulting terms are combined using OR operations.

For example, consider the Boolean function:

$F = A \cdot B + A' \cdot C$

Here, $A \cdot B$ and $A' \cdot C$ are product terms formed using AND operations. These terms are then added using the OR operation. Therefore, this expression is in SOP form.

In the POS form, a Boolean expression is written as a product, which means AND operation, of two or more sum terms. Each sum term is formed by ORing variables or their complements. In this case, first the variables are combined using OR operations, and then the resulting terms are combined using AND operations.

For example, consider the Boolean function:

F = (A + B) (A' + C)

Here, A + B and A' + C are sum terms formed using OR operations. These terms are then multiplied using the AND operation. Therefore, this expression is in POS form.

Thus, SOP form represents a Boolean function as OR of AND terms, while POS form represents it as AND of OR terms, [illegible] and simpli[illegible]

(c) What is a flip-flop ? How is it different

from a latch ? Also, explain the working of the following flip-flops along with necessary diagrams :

(i) Master-Slave flip-flop
(ii) Edge-Triggered flip-flop

A flip flop is a bistable sequential circuit that is capable of storing one bit of binary information. It has two stable states, logic 0 and logic 1, and it remains in one of these states until an external triggering signal causes it to change. Flip flops are important memory elements used in digital systems such as registers, counters, shift registers, and control circuits. They operate with the help of a clock signal, which ensures that changes occur at controlled times.

A latch is also a bistable storage device, but it is level sensitive rather than clock sensitive. This means a latch changes its output whenever the input signal is active and the enable signal is present. In contrast, a flip flop changes its state only at specific moments determined by the clock signal, usually at the rising or falling edge.

Because of this, latches are considered asynchronous devices, while flip flops are synchronous devices. Flip flops provide better timing control, higher reliability, and prevent unwanted changes caused by input fluctuations.

(i) Master Slave Flip Flop

A master slave flip flop is constructed by connecting two latches in series. The first latch is called the master, and the second is called the slave. Both latches are controlled by the same clock signal but operate in opposite phases. When the clock signal is high, the master latch becomes active and accepts the input data, while the slave latch remains inactive. When the clock signal becomes low, the master latch stops accepting input and transfers its stored value to the slave latch, which then updates the output. This arrangement ensures that the output changes only once during each clock cycle. The master slave configuration eliminates race around conditions and provides stable and controlled operation.

The basic block representation can be described as:

Input is applied to the master latch. The master latch passes its stored value to the slave latch depending on the clock signal. The slave latch produces the final output.

(ii) Edge Triggered Flip Flop

An edge triggered flip flop changes its state only at the exact instant of a clock transition, either at the rising edge or falling edge of the clock pulse. It does not respond to input changes during the rest of the clock cycle. This [illegible] circuit that detects the clock edge and allows the input to be stored only [illegible] edge triggered [illegible] when the clock signal [illegible] In a negative edge triggered flip flop, the change occurs when the clock moves from high to low. This design provides precise timing [illegible] one clock [illegible]onous systems.

Thus, flip flops are essential storage elements in digital electronics, and master slave and edge triggered flip flops ensure accurate, stable, and synchronized data storage and processing in computer systems.

3. (a) Assume that a computer system has the following memories :

(i) RAM 64 words with 16-bits word.

RAM of 64 words with 16-bit word means the main memory contains 64 separate storage locations, and each location can store 16 bits of data. Since there are 64 locations, the number of address lines required is 6 because $2^6 = 64$. This means 6 bits are needed to uniquely identify each memory location. The data bus width is 16 bits because each word contains 16 bits that can be transferred at a time. The total storage capacity of this RAM can be calculated as:

Total capacity = Number of words × Bits per word

= 64 × 16 = 1024 bits

This is equal to 128 bytes. Thus, this RAM has 64 addressable locations, requires 6 address lines, uses a 16-bit data bus, and provides a total storage capacity of 1024 bits.

The cache has 8 blocks, and each block size is 32 bits. Since each memory word is 16 bits, one cache block can store 2 words. To find the cache location for decimal address 21 using direct mapping, first determine the main memory block number:

Block number = Address ÷ Words per block

= 21 ÷ 2 = 10 (ignore remainder)

Now apply the direct mapping formula:

Cache location = Block number mod Number of cache blocks

= 10 mod 8 = 2

Therefore, decimal address 21 will be found in cache block location number 2.

(ii) Cache of 8 blocks (block size – 32 bits).

Find in which location of Cache memory a decimal address '21' can be found if

direct mapping is used.

The cache has 8 blocks, and each block size is 32 bits. Since each memory word is 16 bits, one cache block can store 2 words. To find the cache location for decimal address 21 using direct mapping, first determine the main memory block number:

Block number = Address ÷ Words per block

= 21 ÷ 2 = 10 (ignore remainder)

Now apply the direct mapping formula:

Cache location = Block number mod Number of cache blocks

= 10 mod 8 = 2

Therefore, decimal address 21 will be found in cache block l

(b) List t

of Associative Memory.

Associative memory, also known as content addressable memory (CAM), is a special type of memory in which data is accessed based on its content rather than by using a specific memory address. Instead of searching sequentially, associative memory compares the input data with all stored data simultaneously, which makes it very fast.

One major advantage of associative memory is its high speed of data retrieval. Since all memory locations are searched in parallel, the required data can be found almost instantly. This makes it extremely useful in applications where quick searching is necessary, such as cache memory and routing tables. Another advantage is efficient searching capability. It allows direct matching of stored data with input patterns, which reduces search time compared to conventional memory. Associative memory also improves system performance because it reduces the need for multiple memory access cycles. Additionally, it supports flexible data access since information can be retrieved based on partial or complete matching of content.

However, associative memory also has several disadvantages. One major drawback is its high cost. The hardware required for parallel comparison circuits is complex and expensive, making associative memory much costlier than conventional RAM. Another disadvantage is high power consumption, because all memory locations are compared simultaneously during each search operation. It also has limited storage capacity, since building large associative memories is difficult and expensive. Furthermore, the design complexity of associative memory is high, which increases manufacturing difficulty and maintenance requirements.

Thus, associative memory provides very fast and efficient searching, but its high cost, complexity, and power consumption limit its widespread use in large-scale memory systems.

(c) Explain how RAID technology enhances the performance and reliability of data storage. Also, summarize the characteristics of any three RAID-levels.

RAID stands for Redundant Array of Independent Disks. It is a technology used to combine multiple physical hard disks into a single logical unit to improve data storage performance, reliability, and fault tolerance. RAID distributes data across several disks in different ways, depending on the RAID level being used.

RAID enhances performance by allowing parallel data access. Since data is spread across multiple disks, the system can read and write data simultaneously from different drives, which significantly increases speed. This technique is known as data striping. RAID also improves re[illegible] stores extra copies of data or parity information across disks, so if one [illegible] from the remaining disks. This reduces the risk of data loss and increases system availability.

RAID also provides fault tolerance. In case of disk failure, the sy[illegible]ption, and th[illegible]ng the system. [illegible]ance and reliability of data storage systems.

The characteristics of three RAID levels are as follows. RAID 0 uses data striping without redundancy. Data is divided into blocks and stored across multiple disks. It provides very high performance because multiple disks work simultaneously, but it has no fault tolerance. If one disk fails, all data is lost.

RAID 1 uses data mirroring. The same data is stored on two or more disks. This provides high reliability because if one disk fails, the data can be recovered from the mirror disk. However, storage efficiency is low because half of the storage capacity is used for duplication.

RAID 5 uses block level striping with distributed parity. Data and parity information are spread across all disks. It provides a good balance of performance, storage efficiency, and fault tolerance. If one disk fails, data can be reconstructed using parity information.

Thus, RAID technology enhances storage systems by improving speed, ensuring data safety, and providing continuous operation even during hardware failures.

4. (a) Explain the role of the following types of registers :

(i) General purpose register

A general purpose register is a type of processor register that is used to store temporary data during the execution of a program. These registers are directly connected to the arithmetic logic unit (ALU) and are designed to hold operands, intermediate results, and sometimes final results of computations. Because they are located inside the CPU, accessing them is much faster than accessing main memory, which helps improve overall processing speed.

The main role of general purpose registers is to provide fast storage for frequently used data. When a program executes arithmetic or logical operations, the required data is first loaded into these registers. The ALU then performs operations on the data stored in the registers, and the results are also stored back in them. This reduces the number of memory access operations and increases system efficiency.

General purpose registers also support data transfer operations. They act as temporary holding locations when data moves between memory and the processor. For example, before performing an addition operation, data is first fetched from memory into general purpose registers.

In many microprocessors, such as the 8086, examples of general purpose registers include AX, BX, CX, and DX. Each of these can store 16-bit data and can be used for different types of operations, such as arithmetic processing, data storage, and loop counting.

Thus, general purpose registers play a vital role in impr[illegible]ing pr[illegible]ing speed, supporting [illegible]etic operations, and providing temporary storage within the CPU.

(ii) Address Register

An ad[illegible]sed to store the [illegible]ons that the CPU needs to access. It does not store the actual data but holds the location where the data is present in memory. Address registers play a very important role in communication between the processor and main memory.

The primary function of an address register is to specify the address of a memory location during read and write operations. When the CPU wants to fetch an instruction or data from memory, it first places the required memory address into the address register. This address is then sent through the address bus to the memory unit so that the correct location can be accessed.

Address registers are also used in various addressing modes. For example, in indexed addressing and indirect addressing, address registers help in calculating the effective address of an operand. They are also useful in looping and array processing, where the register value can be incremented or decremented to access consecutive memory locations.

A common example of an address register is the Memory Address Register (MAR), which holds the address of the memory location that is currently being accessed by the CPU.

Thus, address registers are essential for locating data and instructions in memory, enabling efficient communication between the processor and memory system.

(iii) Data Register

A data register is a special type of CPU register used to temporarily store data that is being processed, transferred, or fetched from memory. Unlike an address register, which holds memory locations, a data register holds the actual data values. These registers are directly connected to the data bus and the arithmetic logic unit, making them essential for processing operations.

The main role of a data register is to hold data during input, output, and processing operations. When the CPU reads data from memory, the data is first stored in a data register before being sent to the ALU for computation. Similarly, when the CPU writes data to memory, the data is first placed into the data register and then transferred to the specified memory location. Data registers also play an important role in buffering. Since the CPU and memory operate at different speeds, data registers temporarily hold information to ensure smooth and synchronized data transfer. This helps prevent data loss and improves system efficiency.

A common example of a data register is the Memory Data Register (MDR), also known as the Memory Buffer Register (MBR). It stores data that is either being read from memory or written to memory.

Thus, data registers are essential for storing and trans[illegible] the CPU and memory during processing operations.

(b) With the help of a block diagram, explain the fixed-point arithmetic logic unit.

A fixed [illegible] a digital circuit inside the CPU that performs arithmetic and logical operations on fixed point numbers. In fixed point representation, the position of the decimal or binary point is fixed, which means numbers are treated as integers with an assumed scaling factor. This type of ALU is commonly used in processors where fast and efficient integer calculations are required.

The fixed point ALU performs operations such as addition, subtraction, multiplication, division, and logical operations like AND, OR, NOT, and XOR. It works on binary data stored in registers and produces results that are also stored in registers. Since fixed point arithmetic does not require handling of floating decimal positions, the design of the ALU becomes simpler and faster compared to floating point units.

The block diagram of a fixed point ALU mainly consists of the following components: input registers, arithmetic unit, logic unit, shifter, control unit, and output register.
The input registers hold the operands that are to be processed. These operands are usually taken from general purpose registers inside the CPU. The arithmetic unit performs mathematical operations such as

addition, subtraction, multiplication, and division. It contains circuits like adders, subtractors, and multipliers.

The logic unit performs bitwise operations such as AND, OR, NOT, and XOR. These operations are important for decision making, masking, and bit manipulation tasks. The shifter unit is used to shift bits left or right. This operation is helpful in multiplication, division, and scaling of numbers.

The control unit directs the operation of the ALU. It receives control signals from the CPU instruction decoder and decides which operation needs to be performed. It also controls the flow of data between different components.

Finally, the output register stores the result produced by the ALU before sending it to other parts of the CPU or memory.

Thus, a fixed point ALU is an essential part of the processor that performs fast arithmetic and logical operations on fixed point numbers, enabling efficient data processing in digital computer systems.

(c) Explain the process of execution of a micro-program.

A microprogram is a sequence of microinstructions stored in a special memory called control memory. These microinstructions are used to control the internal operations of the CPU. The process of executing these microinstructions to perform a machine level instruction is called microprogram execution.

[redacted]

micro operations such as data transfer between registers, arithmetic operations, and memory access. Once the starting address is loaded, the control memory provides the first microinstruction.

The microinstruction is then placed into the control buffer register, from where the control signals are generated and sent to different parts of the CPU. These control signals activate specific circuits to perform the required micro operations.

After executing one microinstruction, the next microinstruction address is determined. This may be the next sequential address or a branch address depending on the condition specified in the microinstruction. The process continues until all microinstructions required for that machine instruction are executed.

Finally, when the last microinstruction of the microprogram is completed, the control unit returns to the instruction fetch cycle to process the next machine instruction.

Thus, the execution of a microprogram involves fetching microinstructions from control memory, generating control signals, performing micro operations, and sequencing through all steps until the instruction execution is complete.

5. Write short notes on the following :

(a) Indirect Addressing Mode

Indirect addressing mode is a technique used in computer architecture to access data stored in memory. In this mode, the instruction does not contain the actual address of the operand. Instead, it contains the address of a location that holds the effective address of the operand. In simple terms, the instruction first points to a memory location, and that location contains the real address where the data is stored.

The main purpose of indirect addressing is to provide flexibility in accessing memory. It allows programs to handle large amounts of data efficiently without changing instructions frequently. This addressing mode is especially useful when working with arrays, tables, pointers, and dynamic data structures.

The working process of indirect addressing involves two steps. First, the CPU reads the address mentioned in the instruction. Then it goes to that address and fetches the actual effective address of the operand. After that, the CPU accesses the data stored at that effective address. Because of this two step process, indirect addressing is slightly slower than direct addressing, but it offers greater flexibility.

Indirect addressing is widely used in pointer based programming and loop operations. For example, when processing an array, instead of storing the address of each element in the instruction, the program stores a base address in memory, and the CPU accesses elements through indirect addressing.

One m[illegible] that it allows [illegible] If the data location changes, only the pointer value needs to be updated rather than modifying all instructions.

However, its disadvantage is that it requires additional memory access, which increases execution time.

Thus, indirect addressing mode is an important method that improves flexibility, supports dynamic data access, and simplifies memory management in computer systems.

(b) COM Program

A COM program is a simple executable file format used mainly in early operating systems such as DOS. It is one of the simplest types of executable programs and is commonly used for small utility programs and system level tasks. A COM file contains machine code instructions that can be directly executed by the processor without requiring complex loading procedures.

The structure of a COM program is very simple. It does not contain a header, relocation information, or separate code and data segments. Instead, all code, data, and stack are stored within a single segment.

When a COM program is loaded into memory, it is placed at a fixed starting address, usually at offset 100H. This fixed loading address simplifies execution.

COM programs have a maximum size limitation of 64 kilobytes because they operate within a single memory segment. Due to this limitation, they are suitable only for small programs. Despite this restriction, COM programs execute very fast because they require minimal overhead and do not involve complex linking or loading operations.

Another important feature of COM programs is that they use relative addressing rather than absolute addressing. This makes them easier to load and execute since no address adjustment is required.

The main advantages of COM programs include simplicity, fast execution, and low memory overhead. However, they also have disadvantages such as limited size, lack of modular structure, and absence of advanced features like dynamic linking.

Thus, COM programs represent a basic and efficient executable format that was widely used in early computer systems for small and fast running applications.

(c) Dynamic Random Access Memory (DRAM)

Dynamic Random Access Memory, commonly known as DRAM, is a type of semiconductor memory used as the main memory in computer systems. It is called dynamic because it requires continuous refreshing to maintain stored data. DRAM stores each bit of data in a tiny capacitor wi[illegible]

The [illegible]rical charge storage. Each memory cell contains a transistor and a capacitor. The capacitor stores electrical charge to represent binary data. A charged capacitor represents logic 1, while a discharged capacitor represents logic 0. Howev[illegible]dually decre[illegible]eshed periodica[illegible]

DRAM is widely used because it provides high storage density and low cost compared to other types of memory such as SRAM. It can store a large amount of data in a small physical space, making it ideal for main memory applications in computers.

One major advantage of DRAM is its high capacity and low cost per bit. It also consumes less power compared to static memory when storing large volumes of data. However, DRAM has some disadvantages. It is slower than SRAM because it requires refresh cycles, and the refreshing process increases complexity.

DRAM is commonly used in personal computers, servers, and mobile devices as primary memory because it provides a good balance between cost, capacity, and performance.

Thus, DRAM is an essential component of modern computer systems, providing large, economical, and efficient memory storage.

(d) Von Neumann Architecture

DRAM is widely used because it provides high storage density and low cost compared to other types of memory such as SRAM. It can store a large amount of data in a small physical space, making it ideal for main memory applications in computers.

One major advantage of DRAM is its high capacity and low cost per bit. It also consumes less power compared to static memory when storing large volumes of data. However, DRAM has some disadvantages. It is slower than SRAM because it requires refresh cycles, and the refreshing process increases complexity.

DRAM is commonly used in personal computers, servers, and mobile devices as primary memory because it provides a good balance between cost, capacity, and performance.

Thus, DRAM is an essential component of modern computer systems, providing large, economical, and efficient memory storage.

(d) Von Neumann Architecture

Von Neumann architecture is a computer system design model in which both data and program instructions are stored in the same memory. It was proposed by John von Neumann and forms the basis of most modern computer systems.

In this architecture, a computer consists of five main components: the central processing unit, memory unit, input unit, output unit, and system bus. The CPU itself contains the control unit and arithmetic logic unit. The memory stores both instructions and data in the same location, which simplifies system design.

The working of Von Neumann architecture follows the fetch-decode-execute cycle. First, the CPU fetches an instruction from memory. Then the instruction is decoded to determine the required operation. Finally, the CPU executes the instruction and stores the result back in memory.

One m[illegible]licity. Since bo[illegible] the same memory and communication pathways, hardware design becomes less complex and more economical. It also provides flexibility, allowing programs to be easily modified.

However, it has a major limitation known as the Von Neumann bottleneck. Since a single bus is used for both data and instructions, only one transfer can occur at a time, which limits processing speed.

Despite this limitation, Von Neumann architecture remains widely used because of its simplicity, cost effectiveness, and ease of implementation.

Thus, Von Neumann architecture provides a fundamental model for computer organization, forming the foundation of most modern computing systems.