Unit 1: Structure of Computers

Unit at a glance:

Structure of Computer

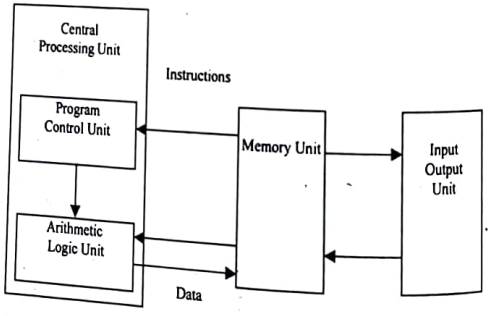
A computer is an electronic device that stores, retrieves and processes data. It can be programmed with instructions and is composed of hardware and software. A computer can exist in a variety of sizes and configurations. There are two types of computers Analog and Digital.

1.1 Von Neumann 'architecture'

Neumann proposed the idea, known as the stored- program concept, which deals with making the programming process easier by representing programs in a form such that they can be suitably stored in memory alongside the data. So, a computer could get its instructions by reading them from memory & also a program could be set or altered depending on the memory values. All of today's computers, generally, having the same general structure & function are thus referred to as ***Von Neumann machines*** and this design is referred to as the ***Von Neumann architecture***.

1.2 Components of Computer system – Structure of CPU, function of Memory unit and IO unit

- Central Processing Unit (CPU)
- Memory Unit
- I/O (input-output) Unit



Structure and Functions of the Different Units of a Digital Computer

(a) Memory Unit

Its purpose is to store both instructions & data. It is also called the Random-Access Memory (RAM) because the CPU can access any memory location at random.

(b) CPU

It acts as the brain of the computer and performs the bulk of data processing operations in a computer. The two main units of a CPU are the Arithmetic Logic Unit and the Program Control Unit. The important parts of CPU are:

(i) Arithmetic Logic Unit (ALU): It performs instructions related to arithmetic operations like ADD, SUB, MUL etc. and logical operations like AND, OR etc.

(ii) Program Control Unit (PCU): It interprets & sequences instructions i.e., interprets & sequences which instruction in a program is to be executed first.

(iii) Register Sets: These are collections of registers that store data.

(c) Input-Output (I/O) Unit

This unit provides an efficient mode of communication between the central system (computer) & the outside environment. Through the I/O unit, programs & data must be entered into computer memory for processing & results obtained from computations must be recorded or displayed to the user.

1.3 Different generation of Computer system

Modern generation computers exist in the following categories:

- Mainframe computer
- Minicomputer
- Microcomputer
- Supercomputer

1.4 Registers

- Computer registers are designated by upper case letters (and optionally followed by digits or letters) to denote the function of the register.
- For example, the register that holds and address for the memory unit usually called a memory address register and is designed by the name **MAR**.
- Other designations for registers are ***PC*** (for program counter), ***IR*** (For instruction register, and ***R1*** (for processor register).
- The individual flip-flops in an n-bit are numbered in sequence from 0 through n-1, starting from 0 in the rightmost position and increasing the numbers toward the left.
- Figure below shows the representation of registers in block diagram form.

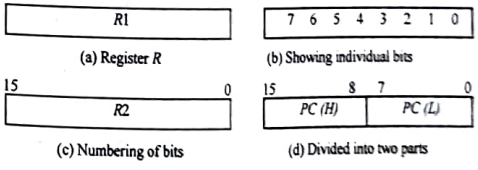


Fig: 4-1 Block diagram of register.

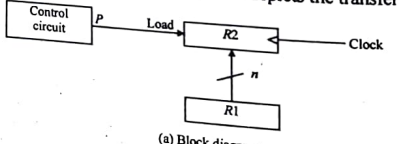
- The most common way to represent a register is by a rectangular box with the name of the register inside, as in Fig. (a).
- The individual bits can be distinguished as in (b).
- The numbering of bits in a 16-bit register can be marked on top of the box as shown in (c).
- 16-bit register is partitioned into two parts in (d). Bit 0 through 7 are assigned the symbol L (for low byte) and bits 8 through 15 are assigned the symbol *H* (for high byte).
- The name of the 16-bit register is *PC*. The symbol *PC (0-7)* or *PC (L)* refers to the low-order byte and PC (8-15) of *PC (H)* to the high-order byte.

1.5 Register Transfer:

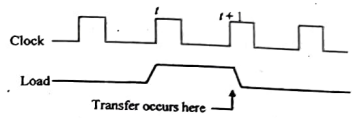
- Information transfer from one register to another is designated in symbolic form by means of a *replacement operator*.
- The statement **R2←R1** denotes a transfer of the content of register R1 into register R2.
- It designates a replacement of the content of R2 by the content of R1.
- By definition, the content of the source register R1 does not change after the transfer.
- If we want the transfer ti occur only under a predetermined control condition then it can be shown by an if-then statement.

  If (P=1) then R2←R1
- P is the control signal generated by a control section.
- We can separate the control variables from the register transfer operator by specifying a ***Control Function***.
- Control function is a Boolean variable that is equal to 0 or 1.
- Control function is included in the statements as

  P: R2←R1
- Control condition is terminated by a colon implies transfer operation be executed by the hardware only if P = 1.
- Every statement written in a register transfer notation implies a hardware construction for implementing the transfer.
- Figure below shows the block diagram that depicts the transfer from R1 to R2.



(a) Block diagram



(b) Timing diagram

Fig: Transfer from R1 to R2 when p = 1.

- The n outputs of register R1 are connected to the n inputs of register R2.
- The letter n will be used to indicate any number of bits for the register. It will be replaced by an actual number when the length of the register is known.
- Register R2 has a load input that is activated by the control variable P.
- It is assumed that the control variable is synchronized with the same clock as the one applied to the register.
- As shown in the timing diagram, P is activated in the control section by rising edge of the pulse at time *t*.
- The next positive transition of the clock at time *t*+1 finds the load input active and the data inputs of R2 are then loaded into the register in parallel.
- P may go back to 0 at time *t*+1; otherwise, the transfer will occur with every clock pulse transition while P remains active.
- Even through the control condition such as P becomes active just after time *t*, the actual transfer does not occur until. The register is triggered by the next positive transition of the clock at time *t*+1.
- The basic symbols of the register transfer notation are listed in below table

Symbol	Description	Examples
Letter (and numerals)	Denotes a register	MAR, R2
Parentheses ( )	Denotes a part of a register	R2(0-7), R2(L)
Arrow <-	Denotes transfer of information	R2 <-R1
Comma ,	Separates two micro operations	R2 <-R1, R1 <R2

- A comma is used to separate two or more operations that are executed at the same time.
- The statement

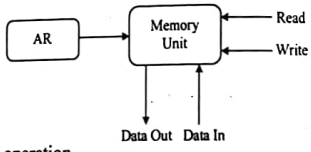
  T: R2←R1, R1←R2 (exchange operation) denotes an operation that exchanges the contents the contents of two registers during one common clock pulse provided that T=1.

1.6 Memory Transfer:

The transfer of information from a memory word to the outside environment is called a read operation. The transfer of new information to be stored into the memory is called a write operation.

Different Registers Associated for Memory Transfer:

The address register (AR) is used to select a memory address, and the data register (DR) is used to send and receive data. Both these registers are connected to the internal bus.



DR← M[AR]: This is the read operation.

M[AR]<--R1: This is the write operation.

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. The three main components of a digital computer system are – **[WBSCTE 2011, 2014, 2015, 2019, 2021]**
   (a) Memory, IO, DMA (b) ALU, CPU, Memory
   (c) Control Unit, ALU, Register (d) CPU, Memory, IO
   Answer: (d)

2. A subtractor is not usually present in a computer because – **[WBSCTE 2011, 2015, 2018]**
   (a) It is expensive (b) It is not possible to design it
   (c) The adder will take care of subtraction (d) None of the above
   Answer: (c)

3. Processors of all computers, whether micro, mini or mainframe must have – **[WBSCTE 2011, 2015]**
   (a) ALU (b) Primary Storage (c) Control unit (d) All of these
   Answer: (d)

4. The ascending order or a data Hierarchy is – **[WBSCTE 2017]**
   (a) bit-bytes-fields-record-file-database (b) bit-bytes-record-field-file-database
   (c) bytes-bit-field-record-file-database (d) bytes-bit-record-field-file-database
   Answer: (a)

5. Overflow occurs when – **[WBSCTE 2012]**
   (a) data is out of range (b) data is within range
   (c) none of these (d) data is greater than upper range
   Answer: (d)

6. Which of the following comment about the program counter is true? **[WBSCTE 2014, 2015]**
   (a) It is used to count the number of instructions
   (b) It is a cell in ROM
   (c) During execution of the current instruction, its content changes
   (d) None of the above
   Answer: (c)

7. Processors of all computers, whether micro, mini or mainframe must have **[WBSCTE 2017]**
   (a) ALU (b) Primary storage (c) Control Unit (d) all of these
   Answer: (d)

8. A typical modern computer uses **[WBSCTE 2018]**
   (a) LSI chip (b) VLSI chip (c) valves (d) vacuum tube
   Answer: (b)

9. The three main components of a digital computer system are **[WBSCTE 2021]**
   (a) Memory, IO, DMA (b) ALU, CPU, Memory
   (c) CU, ALU, Register (d) CPU, Memory, IO
   Answer: (d)

10. The second generation of computer used **[WBSCTE 2021]**
    (a) transistors (b) IC (c) vacuum tube (d) LSI
    Answer: (a)

11. Processors of all computers, whether micro, mini or mainframe must have **[WBSCTE 2021]**
    (a) ALU (b) Primary storage (c) Control Unit (d) all of these
    Answer: (d)

12. What does CSA stands for? **[WBSCTE 2022]**
    (a) Computer Service Architecture (b) Computer Speed Addition
    (c) Carry Save Addition (d) None of these
    Answer: (b)

13. Individual control word of the micro routine are called as **[WBSCTE 2022]**
    (a) Micro task (b) Micro instruction
    (c) Micro operation (d) Micro Command
    Answer: (b)

14. Which of the following circuit convert the binary data into a decimal? [WBSCTE 2022]
(a) Decoder (b) Encoder (c) Code converter (d) Multiplexer
Answer: (c)

15. In full adders the sum circuit is implemented using _____. [WBSCTE 2022]
(a) And & or gates (b) NAND gate
(c) XOR (d) XNOR
Answer: (c)

16. Computer address bus is [WBSCTE 2022]
(a) Unidirectional (b) Bidirectional
(c) Multidirectional (d) None of the above
Answer: (b)

17. Which of the following computer bus connects the CPU to a memory on the system board? [WBSCTE 2022]
(a) Expansion bus (b) Width bus
(c) System bus (d) None of the above
Answer: (c)

18. Micro operation is shown as [WBSCTE 2022]
(a) R1←R2 (b) R1+R2 (c) Both (d) None
Answer: (a)

19. Gray code for decimal 7 is [Model Question]
(a) 0111 (b) 1011 (c) 0100 (d) 0101
Answer: (c)

20. 9's complement of 23 is [Model Question]
(a) 77 (b) 76 (c) 56 (d) 89
Answer: (b)

21. BCD numbers express each decimal digit as [Model Question]
(a) Byte (b) Nibble (c) Bit (d) ASCII
Answer: (b)

22. 2's complement of 1010100 is [Model Question]
(a) 0110011 (b) 0101100 (c) 1010101 (d) 0010010
Answer: (b)

23. The sum of $(24D)_{16}$ and $(9AA)_{16}$ [Model Question]
(a) $(BE7)_{16}$ (b) $(BF6)_{16}$ (c) $(AF7)_{16}$ (d) $(BF7)_{16}$
Answer: (d)

24. The sum of $(10110)_2$ and $(1100)_2$ is [Model Question]
(a) 011011 (b) 100011 (c) 001100 (d) 100010
Answer: (d)

25. 9's complement of 546700 is [Model Question]
(a) 453299 (b) 483270 (c) 32955 (d) 669290
Answer: (a)

26. The 2's complement of 1101100 is [Model Question]
(a) 0010100 (b) 11001100 (c) 11111111 (d) 11110000
Answer: (a)

27. Gray code for decimal 12 is [Model Question]
(a) 1100 (b) 1011 (c) 1010 (d) 0100
Answer: (c)

28. 9's complement of 46 is [Model Question]
(a) 54 (b) 64 (c) 63 (d) 53
Answer: (d)

29. An arithmetic left shift [Model Question]
(a) multiplies a signed number by 2 (b) divides a signed number by 2
c) multiplies a signed number by 4 (d) divides a signed number by 4
Answer: (a)

30. Gray code for decimal 7 is [Model Question]
(a) 0111 (b) 1011 (c) 0100 (d) 0101
Answer: (c)

31. 9's complement of 23 is [Model Question]
(a) 77 (b) 76 (c) 56 (d) 89
Answer: (b)

32. BCD numbers express each decimal digit as [Model Question]
(a) Byte (b) Nibble (c) Bit (d) ASCII
Answer: (b)

33. 2's complement of 1010100 is [Model Question]
(a) 0110011 (b) 0101100 (c) 1010101 (d) 0010010
Answer: (b)

34. The sum of $(24D)_{16}$ and $(9A(A)_{16}$ [Model Question]

(a) $(BE7)_{16}$ (b) $(BF6)_{16}$ (c) $(AF7)_{16}$ (d) $(BF7)_{16}$

Answer: (d)

35. The sum of $(10110)_2$ and $(1100)_2$ is [Model Question]

(a) 011011 (b) 100011 (c) 001100 (d) 100010

Answer: (d)

36. 9's complement of 546700 is [Model Question]

(a) 453299 (b) 483270 (c) 32955 (d) 669290

Answer: (a)

37. The 2's complement of 1101100 is [Model Question]

(a) 0010100 (b) 11001100 (c) 11111111 (d) 11110000

Answer: (a)

38. Gray code for decimal 12 is [Model Question]

(a) 1100 (b) 1011 (c) 1010 (d) 0100

Answer: (c)

39. 9's complement of 46 is [Model Question]

(a) 54 (b) 64 (c) 63 (d) 53

Answer: (d)

40. An arithmetic left shift [Model Question]

(a) multiplies a signed number by 2 (b) divides a signed number by 2

(c) multiplies a signed number by 4 (d) divides a signed number by 4

Answer: (a)

41. The number of multiplexers required to construct a common bus for 4 registers with 8 bits each is [Model Question]

(a) 8 (b) 4 (c) 16 (d) 1

Answer: (a)

42. Masking is synonymous to ..... [Model Question]

(a) OR (b) XOR (c) AND (d) NOT

Answer: (b)

43. The ..... register is used to store result of an instruction. [Model Question]

(a) Program counter (b) Base register

(c) Flag register (d) none of these

Answer: (d)

44. The number of multiplexers required to construct a common bus for 8 registers with 4 bits each is [Model Question]

(a) 16 (b) 8 (c) 4 (d) 2

Answer: (c)

45. **35.** Computer registers are designated by is [Model Question]

(a) capital letters (b) both capital and small letters

(c) numerals (d) small letters

Answer: (a)

46. A logical shift is one that transfers ..... through the serial input. [Model Question]

(a) 0 (b) 1 (c) either 0 or 1 (d) both 0 and 1

Answer: (a)

47. The contents of a Base Register may be changed in _____ mode. [Model Question]

(a) User (b) Privileged (c) Safe (d) none of these

Answer: (a)

48. 8085 has a total of ..... registers. [Model Question]

(a) 10 (b) 11 (c) 12 (d) 13

Answer: (b)

49. ADD is a ..... address instruction. [Model Question]

(a) zero (b) one (c) two (d) three

Answer: (a)

50. ..... calculates the address of the next microinstruction to be executed. [Model Question]

(a) program counter (b) address computation circuit

(c) instruction register (d) none of these

Answer: (a)

51. ..... is connected to the address lines of the system bus. [Model Question]

(a) Instruction register (b) Program counter

(c) Memory address register (d) memory buffer register

Answer: (c)

52. Where the result of an arithmetic and logical operation are stored? [Model Question]

(a) In Accumulator (b) In Cache Memory

(c) In ROM (d) In Instruction Registry

Answer: (a)

53. The number of multiplexer required to construct a common bus for 8 registers with 16 bits each is [Model Question]
(a) 8 (b) 16 (c) 4 (d) 2
Answer: (b)

54. ROR is a [Model Question]
(a) Program control instruction (b) Shift instruction
(c) Logical instruction (d) Data transfer instruction
Answer: (b)

B. Fill in the blanks in the following statements:

55. VLSI stands for ***Very Large Scale Integration***. [WBSCTE 2010]

56. First PC was designed by ***Von-Neumann***. [WBSCTE 2010]

57. Gray Code is also called as ***reflected binary code***. [WBSCTE 2022]

58. Microinstruction consists of ***micro-operations***. [WBSCTE 2022]

59. A source program is usually in ***high-level*** language. [WBSCTE 2022]

60. Loop unrolling is a technique to improve ***performance***. [WBSCTE 2022]

C. Answer the following questions:

61. What is computer bus?
Answer: [WBSCTE 2013]
Computer bus (often simply called **Bus**) is part of some computers. Its role is to transfer data, signals or power between some of the components that make up a computer.
Width: The **size** or **width** of a bus is how many bits it carries in parallel. Common bus sizes are: 4 bits, 8 bits, 12 bits, 16 bits, 24 bits, 32 bits, 64 bits, 80 bits, 96 bits, and 128 bits. Such buses are in wide use:

- Link between the CPU and on-board Memory.
- Link between multiple CPUs in a multi-cpu system
- Link the Arithmetic logic unit to the rest of the CPU
- Connect hard drives, graphics cards, etc. to the main system.
- Serial ATA, ATAPI, USB and Firewire connections

62. Name the CPU registers.
Answer: [WBSCTE 2013]
Actually A Register in CPU terminology is small Named chunk of memory Available inside a microprocessor (CPU), Registers have specific names, sizes and functions varies from processor to processor, for example, if take 8085 microprocessor is an 8-bit

processor which has 8-bit Registers(A:Accumulator, B,C,D,E, H, and L Registers and one flag register all are 8-bit). Two 16-bit registers PC and SP all are having special function and functions comes into the picture during assembly programming. Few registers control is beyond the programmer. If you take another processor Registers will vary let say 8086 is 16-bit processor and it has AX, BX, CX and DX all are 16-bit, PC, SP and Flag registers.

63. How does a computer differ from a calculator? [WBSCTE 2017, 2021]
Answer:
A **computer** is an electronic device which is capable of receiving information and performing a sequence of operations by the instructions given to produce a result in the form of output. A **calculator** is a device used for mathematical calculations and has a small keyboard and display screen that shows the results.

64. What is Bus? [WBSCTE 2021, 2022]
Answer: A bus is a subsystem that is used to connect computer components and transfer data between them.

65. What is super computer? [WBSCTE 2019, 2021]
Answer: ***Refer to Question No 15.a) of Long Answer Type Questions.***

66. What is IO processor? [WBSCTE 2021]
Answer:
The input/output processor or I/O processor is a processor separate from the CPU designed to handle only input/output processes for a device or the computer.

67. What is MAR and MDR? [WBSCTE 2022]
Answer:
Memory Address Register (MAR) - holds the address of the current instruction that is to be fetched from memory, or the address in memory to which data is to be transferred.
Memory Data Register (MDR) - holds the contents found at the address held in the MAR, or data which is to be transferred to primary memory.

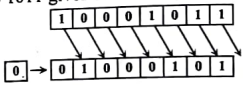
68. What is register? [WBSCTE 2022]
Answer:
A processor register (CPU register) is **one of a small set of data holding places that are part of the computer processor.** A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters).

69. Which is an error-detecting code? [WBSCTE 2022]
Answer:
Error-detecting codes are a sequence of numbers generated by specific procedures for detecting errors in data that has been transmitted over computer networks.

70. What is the logic shift? [WBSCTE 2022]

Answer:

A *logical* shift moves bits to the left or right. The bits whieh 'fall off' the end of the word are discarded and the word is filled with 0's from the opposite end. A logical right shift of the 8 bit binary number 1000 1011 gives 0100 0101, as shown below:



71. What type of device converts digital signal into a form that is intelligible to the user? [WBSCTE 2022]

Answer:

Output devices converts digital signal into a form that is intelligible.

72. Define clock rate? [WBSCTE 2022]

Answer:

The clock rate, or clock speed, of a computer is the rate at which a central processing unit (CPU) is able to perform basic functions.

73. What is clock signal in COA? [WBSCTE 2022]

Answer:

Clock signal is a periodic signal and its ON time and OFF time need not be the same. You can represent the clock signal as a square wave, when both its ON time and OFF time are same.

74. Is USB is a bus? [WBSCTE 2022]

Answer:

A Universal Serial Bus (USB) is a common interface that enables communication between devices and a host controller such as a personal computer (PC) or smartphone.

Long Answer Type Questions

1. Discuss the difference of micro and mainframe computers. [WBSCTE 2004, 2005, 2006, 2007]

OR,

What are the differences between mainframe and personal computer? [WBSCTE 2008, 2010]

Answer:

MICROCOMPUTERS	MAINFRAME COMPUTERS
These are the smallest range of computers generally used for personal usage.	These are large, powerful, and expensive computers used mainly by large companies.

MICROCOMPUTERS	MAINFRAME COMPUTERS
They are usually used for small range of computations.	They are usually used for bulk data processing like financial transaction processing.
Microcomputers are inferior to mainframes in terms of speed and efficiency.	Mainframes have high speed, massive internal memory, high-capacity external storage and fast input output operations.
Desktop PC is an example of microcomputer.	Commercially used mainframes are IBM's z/OS, OS/390.

2. Explain the development of different generations of computer. [WBSCTE 2004, 2005, 2010]

OR,

Describe the different generation of computer. [WBSCTE 2015]

Answer:

There are five computer generations:

1) First generation (1942-1955): The first generation of computer used vacuum tubes as hardware components.

Advantages

- Vacuum tubes were the only electronic components available during those days.
- These computers were the fastest calculating devices of their time. They could perform computation in milliseconds.

Disadvantages

- Large in size i.e., requires more space.
- Thousands of vacuum tubes were used which produced large amount of heat and hardware failures.
- Air conditioning required.
- Not easily portable because of large size.
- Limited commercial use.
- High cost.
- Unreliable.

2) Second generation (1955-1964): The second generation of computer used transistor as a hardware component. The transistor, a smaller and more reliable successor to the vacuum tube, was invented in 1947.

Advantages

- Smaller in size as compared to first generation computer.
- Less heat generated.
- Computation time is much faster than the First generation computers
- Better portability.
- Became commercially popular.
- Better reliability.

Disadvantages

- Air-conditioning required.
- Commercial production was difficult and costly.
- Frequent maintenance required.

3) Third generation (1964-1975): The third generation of computer used integrated circuit (1C) as a hardware instead of transistor, resistor, capacitor across one single chip of silicon.

Advantages

- Smaller in size
- More reliable
- Less heat generated
- Reduce computational time from microseconds to nanoseconds
- Maintenance cost is low because hardware failures are rare
- Easily portable
- Totally general purpose; widely used for various commercial applications.

Disadvantage

- Air-conditioning is required in many cases.

4) Fourth generation (1975 onwards): The fourth generation of computer used Small Scale Integration (SSI), Large Scale Integration (LSI) and Very Large Scale Integration (VLSI) as a hardware component.

Advantages

- Smaller in size.
- Cost effective.
- Heat generation is negligible.
- Air-conditioning is not required in most cases.
- Much faster than it's predecessors.
- Easily portable because of small size.
- For general-purpose use.

Disadvantage

- Highly sophisticated technology required for manufacture of LSI chips. Repairing cost is high.

5) Fifth generation (Yet to come): Scientists are now working on the fifth generation of computer. They aim to bring us machines with genuine I.Q., the ability to reason logically and with real knowledge of the world. Fifth generation will be totally different, totally novel, and totally new.

➲ 3. What are address, data and control bus? **[WBSCTE 2006, 2007, 2009, 2010]**

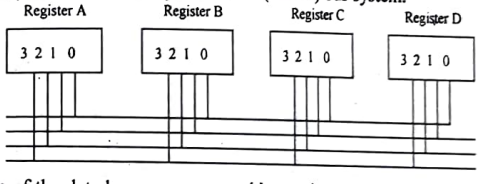
Answer:

Data Lines: These lines provide a path for moving data between registers. These are designed to transmit all bits of an n-bit word in parallel. So, they either consist of two sets of n-unidirectional lines or a single set of n-bi-directional lines.

Data lines are collectively called the ***data bus***. This bus of size 'n' is usually a multiple of 8, with n = 8, 16 or 32 etc. separate lines. Since each line can carry only 1 bit at a time, the number of lines determines how many bits can be transferred at a time.

Example with Diagram

Let us now consider an example. In fig, there are four registers A, B, C and D with four bits (3, 2, 1, 0) each connected by a common (shared) bus system.



As each line of the data bus can carry one bit at a time, so the data bus must have ***either*** two sets of 4 unidirectional lines [i.e., 2*(n=4) = 8] or 8 lines, out of which one set (i.e., 4 lines) will transmit data in the forward direction & the other set will transmit data in the reverse direction. ***Otherwise***, the data bus will have 4 (i.e., 1 set) bi-directional lines. Number of bi-directional lines in the data bus = number of bits in each register.

Address Lines: Address lines of the bus are used to transfer the addresses of the source register sending the data and the destination register receiving the data. Address lines are collectively called the ***address bus***.

Example

Consider Fig. Suppose register A is sending data to register C. So via the address lines of the bus, both the sender's address (i.e. register A) and the receiver's address (i.e. register C) are transmitted to let all the registers sharing the bus identify the sender as well as the receiver of the data.

Control Lines: These lines, transmitting control signals, are used to control the access to the data & address lines i.e. control lines select which register, out of the multiple registers, will transmit its information through the bus.

Also control lines are used to transfer timing signals & status information about the registers in the system as well as to indicate the type of information present on the data lines.

Since all registers share the data & address lines, there must be means of controlling their use, which is done by the control lines.

Example

Figure shows the control lines of the shared bus among the 4 registers.

4. What are the different components of a CPU? Explain with suitable diagram. **[WBSCTE 2007, 2009, 2010(Short note)]**

OR,

Draw the block diagram of CPU and explain the function of each part neatly. **[WBSCTE 2011]**

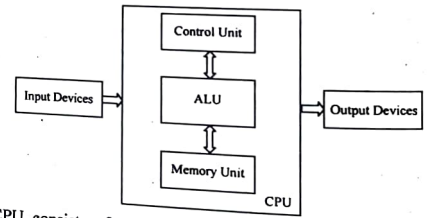
OR,

Draw the block diagram of CPU and explain the execution of typical instruction through different section of CPU. **[WBSCTE 2015]**

Answer:

The "heart" of a computer system is the central processing unit (or CPU) - the part where the actual processing of data takes place. It consists of three separate sub-parts, namely

- The main memory or store (where instructions and data are kept in coded form)
- The arithmetic/logic unit (ALU) contains the electronic circuitry that executes all arithmetic and logical operations like addition, subtraction, multiplication, and division and logical operation is usually a comparison. The unit can compare numbers, letters, or special characters. The computer can then take action based on the result of the comparison.
- A computer can simultaneously test for more than one condition. In fact, a logic unit can perform six logical relationships: equal to, less than, greater than, less than or equal to, greater than or equal to, and not equal.
- The control unit of the CPU contains circuitry that uses electrical signals to direct the entire computer system to carry out, or execute, stored program instructions. The control unit does not execute program instructions, rather, it directs other parts of the system to do so. The control unit must communicate with both the arithmetic/logic unit and memory.



Working:

- CPU consists of three basic units: control unit, Arithmetic Logical Unit (ALU) and memory unit.
- Input is given through the input devices to CPU.
- Control unit controls communication within ALU and memory unit.
- Decides which circuit is to be activated.
- For reading instruction it uses Fetch-execute mechanism.
- Control unit gets instruction from memory.

- Control unit decides what to do of that instruction and transfers it to the ALU.
- ALU performs various arithmetic operations like addition, subtraction, multiplication, division and logical operations like AND, OR, NOT, NAND etc. on that instruction.
- Results of ALU are stored in the memory or resistor for its further operations.
- After completing the instruction, stored results are passed to the output devices.
- To synchronize all these operations CPU uses its own system clock.

Functions:

- Executes stored instructions called as program.
- Tells rest of the computer system what to do.
- Executes arithmetic calculation and data manipulation.
- Holds data and instruction which are in the current use.
- Responsible for storing and retrieving information on disks and other media.

5. (a) What is meant by multiplexing of data bus and address bus? **[WBSCTE 2008, 2010]**

(b) What is the function of control bus? **[WBSCTE 2008]**

Answer:

a) Sometimes data lines may be used to transfer data as well as addresses. This is termed as data/address multiplexing. This may be done to decrease the cost of the bus (i.e. less number of cables are required) and also to decrease the number of external connections (pins). This concept is generally common in case of data transfers via an I/O bus i.e. while the CPU is interfacing with the I/O units but not used in case of memory interfacing.

b) These lines, transmitting control signals, are used to control the access to the data & address lines i.e. control lines select which register, out of the multiple registers, will transmit its information through the bus.

6. Write major features of Micro-computer, Mainframe and Supercomputers. **[WBSCTE 2009]**

Answer:

Microcomputer: These are the smallest range of computers in terms of speed and efficiency. Today's desktop PCs are known as microcomputers. They are usually used for small range of computations in comparison to mainframes and minicomputers.

Mainframe computer: These are large, powerful, and expensive computers used mainly by large companies for bulk data processing like financial transaction processing. Mainframes have high speed, massive internal memory, high-capacity external storage and fast input output operations. Commercially used mainframes are IBM's z/OS, OS/390.

Supercomputer: These are the fastest type of computer. Supercomputers are very expensive and are employed for specialized applications that require large complex mathematical calculations. For example, weather forecasting requires a supercomputer. Other uses of supercomputers include animated graphics, fluid dynamic calculations, nuclear energy research, and so on.

7. Differentiate PC/XT and PC/AT. [WBSCTE 2010, 2011]

Answer:

PC/XT	PC/AT
The IBM Personal Computer XT, often shortened to the IBM XT, PC XT, or simply XT, was IBM's successor to the original IBM PC, factory equipped with a hard drive.	The IBM Personal Computer AT, more commonly known as the IBM AT and also sometimes called the PC AT or PC/AT, was IBM's second-generation PC, designed around the 6 MHz.
XT stands for X-tended Technology.	The name AT stood for "Advanced Technology"
Operating system: IBM BASIC / PC DOS 2.0-3.20 / SCOXenix	Operating system: PC-DOS 3.0 and later, OS/2 1.x
CPU: Intel 8088 @ 4.77 MHz	CPU: Intel 80286 @ 6 and 8 MHz
Memory: 128-640 kB	Memory: 256 KB ~ 16 MB

8. What do you mean by stored program concept? Who introduced it?

Answer: [WBSCTE 2010]

This describes a design architecture for an electronic digital computer with subdivisions of a processing unit consisting of an arithmetic logic unit and processor registers, a control unit containing an instruction register and program counter, a memory to store both data and instructions, external mass storage, and input and output mechanisms. The meaning of the term has evolved to mean a stored-program computer in which an instruction fetch and a data operation cannot occur at the same time because they share a common bus. It can be summarized as follows —

1. **Programs are stored in memory**
   - Instructions are also bits
   - To be read or written just like data
2. **Fetch & execute cycle**
   - Instructions are fetched and put into a special register
   - Bits in the register "control" the subsequent actions
   - Fetch the "next" instruction and continue
3. **Instructions**
   - Language of the machine
   - Primitive than higher level languages; e.g., no sophisticated control flow
   - Very restrictive (e.g., MIPS Arithmetic Instructions)

- We'll be working with the MIPS instruction set architecture

Von Neumann introduced the stored program concept.

9. Discuss Charles Babbage's Concept in details. [WBSCTE 2010]

Answer:

Charles Babbage (1791-1871), computer pioneer, designed two classes of engine, Difference Engines, and Analytical Engines. Difference engines are so called because of the mathematical principle on which they are based, namely, the method of finite differences. The beauty of the method is that it uses only arithmetical addition and removes the need for multiplication and division which are more difficult to implement mechanically. Difference engines are strictly calculators. They cannot be used for general arithmetical calculation. The Analytical Engine is much more than a calculator and marks the progression from the mechanized arithmetic of calculation to fully-fledged general-purpose computation. The Analytical Engine has many essential features found in the modern digital computer. It was programmable using punched cards, an idea borrowed from the Jacquard loom used for weaving complex patterns in textiles. The Engine had a 'Store' where numbers and intermediate results could be held, and a separate 'Mill' where the arithmetic processing was performed. It had an internal repertoire of the four arithmetical functions and could perform direct multiplication and division. It was also capable of functions for which we have modern names: conditional branching, looping (iteration), microprogramming, parallel processing, iteration, latching, polling, and pulse-shaping, amongst others, though Babbage nowhere used these terms. It had a variety of outputs including hardcopy printout, punched cards, graph plotting. The logical structure of the Analytical Engine was essentially the same as that which has dominated computer design in the electronic era - the separation of the memory (the 'Store') from the central processor (the 'Mill'), serial operation using a 'fetch-execute cycle', and facilities for inputting and outputting data and instructions. Thus, Charles Babbage considered a "father of the computer".

10. Draw the block diagram of Von Neumann computer architecture and explain the function of each part neatly. [WBSCTE 2012, 2018]

Answer:

Structure of the Computer Deigned by Neumann

The computer designed based on the idea of stored-program concept proposed by John Von Neumann a mathematician, known as the ***IAS computer.*** It was designed at the Princeton Institute for Advanced studies in 1952.

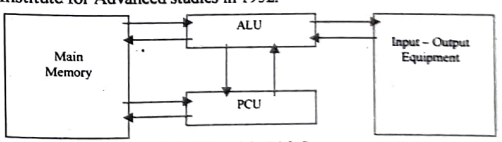


Fig: Structure of the IAS Computer

Main Features of the IAS Computer

The main features of an IAS computer are:

a) A main memory, which stores both data & instructions.
b) An arithmetic-logical unit (ALU) capable of operating on binary data.
c) A control unit, which interprets the instructions in memory & causes them to be executed.
d) Input & output (I/O) unit operated by the control unit.
e) The memory is read-write in nature.
f) The contents of this memory are addressable by location.
g) Execution generally occurs in a sequential fashion from one instruction to the next.

Von Neumann 'architecture'

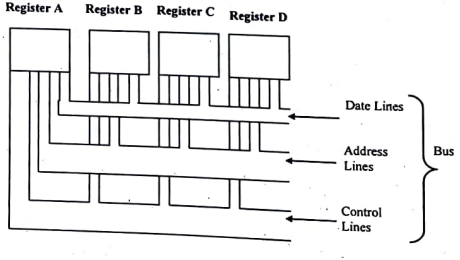
All of today's computers, generally, having the same general structure & function are thus referred to as ***Von Neumann machines*** and this design is referred to as the ***Von Neumann architecture***.

➲ **11. What is bus? How many buses are present in computer system? Describe each of these buses briefly.** **[WBSCTE 2012]**

Answer:

1st Part:

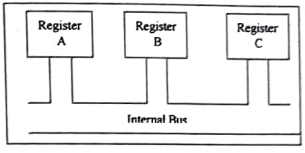
A ***bus*** is a set of wires/cables designed to transfer all bits of a w-bit word from a specified source to a specified destination. The source & destination are typically registers. So, a bus is a communication pathway connecting two or more registers within the system modules namely CPU, Memory, I/O etc. A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one bit at a time. A bus may be unidirectional i.e. capable of transmitting data in one direction only, or it may be bi-directional i.e. capable of transmitting data in both direction.



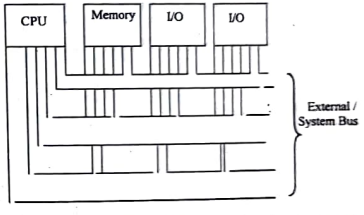
2nd Part:

Bus can be of 4 types:

Internal bus: This runs within the CPU. Internal bus connects all the registers within the CPU.



External / System bus: This runs outside the CPU i.e. the external bus connects all the three subsystems (i.e. connects all the registers within the CPU, Memory and I/O unit) of a digital computer. This is a set of shared communication lines via which the registers inside the Input-Output processors & the CPU share a common access path to main Memory registers.



Shared Bus: Here several registers can be connected via a single common bus with the restriction that only one register can send data at any time.

So, here the number of buses and thus the number of cables required are much less. Cost of cables required, hence, is much less. However, shared buses do not permit simultaneous data transfers between different pair of registers, so this leads to a loss of performance compared to dedicated buses. To implement shared buses, more complicated logic circuits are needed.

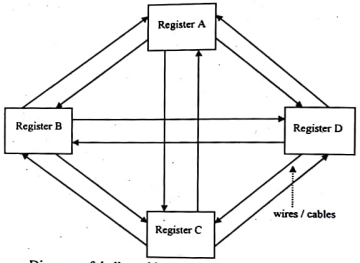


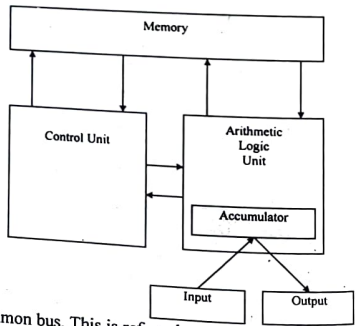
Diagram of dedicated buses connecting the registers

Dedicated Bus: It has a unique source and a unique destination i.e. within each pair of registers there is a pair of dedicated bus for both way transmission of information.
If 'n' registers are interconnected by buses in all possible ways, then the number of dedicated buses required is 'n (n-1)'.
Here the simultaneous transfer of information between different pair of devices is possible.

➲ **12. Discuss the Von Neumann concept in details. Compare Von Neumann architecture with modern computer architecture.** **[WBSCTE 2013]**

Answer:

The term Von Neumann architecture, also known as the Von Neumann model or the Princeton architecture, This describes a design architecture for an electronic digital computer with subdivisions of a processing unit consisting of an arithmetic logic unit and processor registers, a control unit containing an instruction register and program counter, a memory to store both data and instructions, external mass storage, and input and output mechanisms. The meaning of the term has evolved to mean a stored-program computer in which an instruction fetch and a data operation cannot occur at the same time because they share a common bus. This is referred to as the Von Neumann bottleneck and often limits the performance of the system.



➲ **13. a) Explain the representation of floating point number in computer. Describe the IEEE format also.**
b) What is NaN in floating point number representation? Give example.
[WBSCTE 2019]

Answer:

a) 1st Part:

Floating-Point Method: Such a representation uses a second register to store the number that designates the position of the decimal point stored in the first register.
There are two parts in the floating-point representation of a number. They are:
Mantissa: This is the first part. This part is represented by a signed, fixed-point number. Mantissa may be a fraction or it may be an integer.
Exponent: This is the second part, which indicates the position of the decimal point in the number i.e., where the decimal point is located. In this type of representation

$$7.03\times10000000000000000000000=7.03\times10^{23}$$

Consider the number 703000000000000000000000. Eliminating the zeros it can be noted that this number is just

$$7.03\times10000000000000000000000=7.03\times10^{23}$$

This is known as scientific notation.
There are two parts in the floating-point representation of a number. They are:
Mantissa: This is the first part. This part is represented by a signed, fixed-point number. Mantissa may be a fraction or it may be an integer.
Exponent: This is the second part, which indicates the position of the decimal point in the number i.e. where the decimal point is located.
To represent the number +7154.232. The representation of this number in floating-point is as follows:
+0.7154232 (Fraction part or the mantissa part → m)
+04 (exponent part → e)
The exponent part indicates the actual position of the decimal. Here +04 indicates that the actual position of the decimal point in the number is four positions to the right of the decimal point as indicated in the fraction. So this number can be represented as $m\times r^e$ ($+0.7154232\times10^{+4}$), where r is the radix. The m and e values along with their sign-bits are physically represented in registers. Floating-point binary numbers can also be represented similarly.

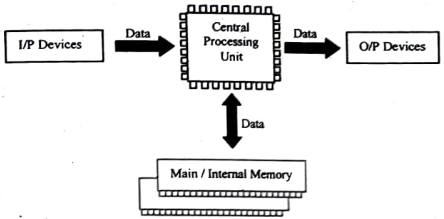
2nd Part:

Exponents are commonly stored in different IEEE standard formats as unsigned integers. However, an exponent can be negative as well as positive, and so there must be some technique for representing negative exponents using unsigned integers. This technique is called "biasing". A positive number is added to the exponent before it is stored in to the floating point number. The stored exponent is then called a "***biased exponent***" and the representation is known as "***biased representation***". For single precision exponent contains 8 bits, the bias number 127 is added to the exponent before it is stored so that, for example, an exponent of 1 is stored as 128. Since the unsigned exponent can represent numbers between 0 and 255, it should be possible to store exponents whose values range

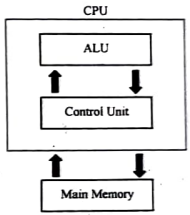
from –127 to +128 i.e., –127 would be stored as the biased exponent value 0, and +128 would be stored as the biased value 255.
The range of actual exponents represented is still the *same*. With the biased exponent, the value represented by the notation is: $(-1)^S \times (1.+M) \times 2^{E\text{-}Bias}$. That is why biased representation is used.
The IEEE 754 standard specifies a double precision as having:
1 sign bit, 11bits Exponent, 53 bits (52 explicitly stored) Significant precision.

b) The act of reaching an invalid result is called a floating-point exception. An exceptional result is represented by a special code called a NaN, for "Not a Number". All NaNs in IEEE 754-1985 have this format: sign = either 0 or 1.
The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and reduced their portability. IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macs, and most Unix platforms.
There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components:
The Sign of Mantissa –This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
The Biased exponent –The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.
The Normalised Mantissa –The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. 0 and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.

➲ **14. a) Explain the components of the Computer system and what is micro operation?**
b) Describe the Von- Neumann Architecture with diagram? Explain the Bus Structure with examples.
Answer: **[WBSCTE 2022]**
a) 1st Part:
Computer systems consist of three components as shown in below image: **Central Processing Unit, Input devices and Output devices**. Input devices provide data input to processor, which processes data and generates useful information that's displayed to the user through output devices. This is stored in computer's memory.



Central Processing Unit
The Central Processing Unit (CPU) is called "the brain of computer" as it controls operation of all parts of computer. It consists of two components: Arithmetic Logic Unit (ALU), and Control Unit.



Arithmetic Logic Unit (ALU)
Data entered into computer is sent to RAM, from where it is then sent to ALU, where rest of data processing takes place. All types of processing, such as comparisons, decision-making and processing of non-numeric information takes place here and once again data is moved to RAM.
Control Unit
As name indicates, this part of CPU extracts instructions, performs execution, maintains and directs operations of entire system.
Functions of Control Unit
Control unit performs following functions –
- It controls all activities of computer
- Supervises flow of data within CPU
- Directs flow of data within CPU
- Transfers data to Arithmetic and Logic Unit
- Transfers results to memory
- Fetches results from memory to output devices

Memory Unit

This is unit in which data and instructions given to computer as well as results given by computer are stored. Unit of memory is "Byte".

1 Byte = 8 Bits

2nd Part:

A microoperation is an elementary operation performed with the data stored in registers.

1) Register transfer microoperation transfer binary information from one register to another.
2) Arithmetic microoperations perform arithmetic operation on numeric data stored in registers.
3) Logic micro operation performs bit manipulation operation on numeric data stored in register.
4) Shift microoperation performs shit operation on data stored in registers.

b) 1st Part:

Most of today's computer designs are based on concepts developed by John von Neumann referred to as the Von Neumann architecture. Von Neumann proposed that their should be a unit performing arithmetic and logical operation on the data. This unit is termed as Arithmetic Logical Unit (ALU). A control unit directs the ALU to perform specific arithmetic or logic function on the data. Therefore in such a system, by changing the control signal the desired operation can be performed on data. Main components of a von Neumann computer Architecture or block diagram of a computer is:

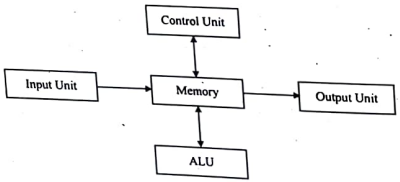


Fig: Block diagram of a computer

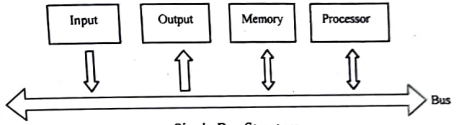
The Arithmetic Logical Unit (ALU) and Control Unit (CU) together are termed as the Central Processing Unit (CPU). The CPU is the most important components of a computer's hardware. The ALU performs the arithmetic operations such as addition, subtraction, multiplication and division, and the logical operations such as: "Is A > B?

How can the instructions and stat be put into the computers? The instruction and data need to be supplied by external devices known as input unit like keyboard. Main responsibility of input unit will be to put the data in the form of signals that can be recognized by the computer. Similarly, we need another component that will show the

results in proper format and form. This component is known as output unit like monitor. These components are referred together as input/output (I/O) components.

2nd Part:

Bus structures in computer plays important role in connecting the internal components of the computer. The bus in the computer is the *shared transmission medium*. This means multiple components or devices use the same bus structure to transmit the information signals to each other. At a time only one pair of devices can use this bus to communicate with each other successfully. If multiple devices transmit the information signal over the bus at the same time the signals overlap each other and get jumbled.



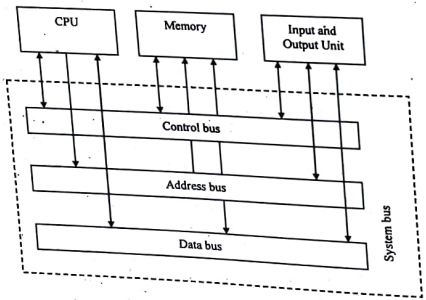
Single Bus Structure

A system bus has typically from fifty to hundreds of distinct lines where each line is meant for a certain function. These lines can be categorises into three functional groups i.e., data line, address lines, and control lines.

1. **Data Lines:** Data lines coordinate in transferring the data among the system components. The data lines are collectively called data bus. A data bus may have 32 lines, 64 lines, 128 lines, or even more lines. The number of lines present in the data bus defines the *width* of the data bus.
   Each data line is able to transfer only one bit at a time. So the number of data lines in a data bus determines how many bits it can transfer at a time. The performance of the system also depends on the width of the data bus.
2. **Address Lines:** The content of the address lines of the bus determines the source or destination of the data present on the data bus. The number of address lines together is referred to as address bus. The number of address lines in the address bus determines its *width*.
   The width of the address bus determines the memory capacity of the system. The dcontent5 of address is also used for addressing I/O ports. The higher-order bits determine the address of memory locations or I/O ports.
3. **Control Lines:** The address lines and data lines are shared by all the components of the system so there must some means to control lines control the use and access to address and data lines of the bus. The control signal consists of the *command* and *timing information*. Here the command in the control signal specify the operation that has to be performed. And the timing information over the control signals specify till when the data and address information is valid.

The control lines include the line for:

- **Memory Write:** This command causes the data on the data bus to be places over the addressed memory location.
- **Memory Read:** This command causes the data on the addressed location to be places on the data bus.
- **I/O Write:** The command over this control line causes the data on the data bus to be placed over the addressed I/O port.
- **I/O Read:** The command over this control line causes the data from the addressed I/O port to be placed over the data bus.
- **Transfer ACK:** This control line indicates the data has been received from the data bus or is placed over the data bus.
- **Bus Request:** This control line indicates that the component has requested control over the bus.
- **Bus Grant:** This control line indicates that the bus has been granted to the requesting component.
- **Interrupt Request:** This control line indicates that interrupts are pending.
- **Interrupt ACK:** This control line provides acknowledgement when the pending interrupt is serviced.
- **Clock:** This control line is used to synchronize the operations.
- **Reset:** The bit information issued over this control line initializes all the modules.



15. Write short notes on the following:

(a) Super computer	**[WBSCTE 2005, 2008, 2010, 2015, 2021]**
(b) Von Neumann Architecture	**[WBSCTE 2007, 2009]**
(c) Control Unit	**[WBSCTE 2008, 2010]**
(d) ALU	**[WBSCTE 2008, 2009, 2010, 2018]**
(e) PC-XT and PC-AT	**[WBSCTE 2012]**
(f) Generation of Computer	**[WBSCTE 2019]**
(g) I/O processor	**[WBSCTE 2021]**
(h) Arithmatic micro-operation	**[Model Question]**
(i) Logic micro-operation	**[Model Question]**
(j) Shift micro-operation	**[Model Question]**

Answer:

a) Super computer:

Super computers are the fastest type of computer. Supercomputers are very expensive and are employed for specialized applications that require large complex mathematical calculations. Supercomputers were introduced in the 1960s and were designed primarily by Seymour Cray at Control Data Corporation (CDC), and later at Cray Research.

Supercomputers are used for highly calculation-intensive tasks such as problems including quantum physics, weather forecasting, climate research, oil and gas exploration, molecular modeling (computing the structures and properties of chemical compounds, biological macromolecules, polymers, and crystals), and physical simulations (such as simulation of airplanes in wind tunnels, simulation of the detonation of nuclear weapons, and research into nuclear fusion).

b) Von Neumann Architecture:

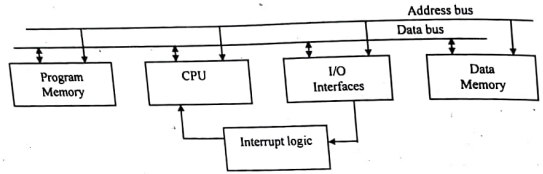
John Von Neumann was a mathematician who was a consultant on the ENIAC project (Electronic Numerical Integrator and Computer), the world's first general-purpose electronic digital computer.

Explanation of the Von Neumann Concept

Neumann proposed the idea, known as the stored- program concept, which deals with making the programming process easier by representing programs in a form such that they can be suitably stored in memory alongside the data. So, a computer could get its instructions by reading them from memory & also a program could be set or altered depending on the memory values.

Von Neumann 'architecture'

All of today's computers, generally, having the same general structure & function are thus referred to as ***Von Neumann machines*** and this design is referred to as the ***Von Neumann architecture***.



c) Control Unit:

The control unit coordinates the components of a computer system. It fetches the code of all of the instructions in the program. It directs the operation of the other units by providing timing and control signals. All computer resources are managed by the control unit. It directs the flow of data between the Central Processing Unit (CPU) and the other devices.

- The control unit does not execute program instructions; rather, it directs other parts of the system to do so. The control unit must communicate with both the arithmetic/ logic unit and memory.

The following figure shows the central processing unit of a computer and how its three basic sub-parts are connected to one another.

d) ALU:

An arithmetic logic unit (ALU) is a major component of the central processing unit of a computer system. It does all processes related to arithmetic and logic operations that need to be done on instruction words. In some microprocessor architectures, the ALU is divided into the arithmetic unit (AU) and the logic unit (LU).

An ALU can be designed by engineers to calculate any operation. As the operations become more complex, the ALU also becomes more expensive, takes up more space in the CPU and dissipates more heat. That is why engineers make the ALU powerful enough to ensure that the CPU is also powerful and fast, but not so complex as to become prohibitive in terms of cost and other disadvantages.

An arithmetic logic unit is also known as an integer unit (IU).

e) PC-XT:

The "IBM Personal Computer XT", IBM's model 5160, was an enhanced machine that was designed for diskette and hard drive storage, introduced two years after the introduction of the "IBM Personal Computer". XT stands for X-tended Technology. It had eight expansion slots and a 10 -20 MB hard disk. The processor was a 4.77 MHz Intel 8088 and the expansion bus 8-bit XT bus architecture.

PC-AT:

The "IBM Personal Computer/AT", IBM's model 5170, used an Intel 80286 processor, originally running at 6- 8 MHz. It had a 16-bit ISA bus and 20 - 30 MB hard drive. The name AT stood for "Advanced Technology". The AT was designed to support

multitasking. IBM PC/ATs were used as more powerful DOS (single-tasking) personal computers.

f) Generation of Computer

Refer to Question No. 2 of Long Answer Type Questions.

g) I/O processor

The Input Output Processor (IOP) is just like a CPU that handles the details of I/O operations. It is more equipped with facilities than those are available in typical direct memory access controller. The IOP can fetch and execute its own instructions that are specifically designed to characterize I/O transfers. In addition to the I/O – related tasks, it can perform other processing tasks like arithmetic, logic, branching and code translation. The main memory unit takes the pivotal role. It communicates with processor by the means of direct memory access.

h) Arithmatic micro-operation

Arithmetic micro operations perform arithmetic operations on numeric data stored in registers.

The basic arithmetic micro operations are additions, subtraction, increment, decrement, and shift. Arithmetic shifts are explained later in conjunction with the shift micro operations. The arithmetic micro operation defined by the statement.

$R3 \leftarrow R1+R2$

It specifies and add micro operation. It states that the contents of register R1 are added to the contents of register R2 and the sum transferred to register R3. To implement this statement with hardware we need three registers and the digital component that performs the addition operation. The other besic arithmetic micro operations are listed in Table.

Subtraction is most often implemented through complementation and addition. Instead of using the minus operator, we can specify the subtraction by the following statement:

$R3 \leftarrow R1+R2'+1$

R2' is the 1's complement of R2, Adding 1 to 1's Complement procedures the 2's Complement form. Adding the R1 to the 2's complement of R2 is equivalent to R1-R2.

Symbolic designation	Description
$R3 \leftarrow R1+R2$	Contents of R1 plus R2 transferred to R3
$R3 \leftarrow R1-R2$	Contents of R1 minus R2 transferred to R3
$R2 \leftarrow \overline{R2}$	Complement the contents of R2 (1's complement)
$R2 \leftarrow \overline{R2}+1$	2's complement the contents of R2 (negate)
$R3 \leftarrow R1+\overline{R2}+1$	R1 plus the 2's complement of R2 (Subtraction)
$R1 \leftarrow R1+1$	Increment the contents of R1 by one
$R1 \leftarrow R1-1$	Determent the contents of R1 by one

The increment micro operation is symbolized by plus one. The decrement micro operation is symbolized by minus 1. These two micro operations are implemented with the help of combinational circuit.

Multiplication and division operations are valid arithmetic operations but are not included in the basic set of micro operations so these are not specified in the above table. Division is implemented with a sequence of subtract and shift micro operations. The multiplication operation is implemented with a sequence of add and shift micro operations. The only place where these operations can be considered as micro operations is in a digital system, where they are implemented by means of a combinational circuit.

i) Another type of micro operation is the logical operation. These operations are concerned with implementing Boolean operations. The common logical operations are given below.

1. AND: $X \leftarrow X \| Y$
2. OR: $X \leftarrow X \vee Y$
3. NOT: $X \leftarrow \overline{X}$
4. XOR: $X \leftarrow X \oplus Y$

On the surface, these look like the standard Boolean operations, until you realize that their operands are multi-bit. For example, each of the registers, X and Y might be 8-bit registers. And, the question then arises as to what the result is when performing an AND operation with two 8-bit operands.

To work with multi-bit operands, processors typically perform *bitwise* Boolean operations. Bitwise operations perform the operation on the corresponding bits of the two operands, without consideration of the other bits. As an example, suppose that you are ANDing two 4-bit values; $0110 \| 0101$. To perform a bitwise AND, you first AND the Position 0 Bits (the low-order bits), independently of the other bits, with the result of $0 \| 1 = 0$. You then AND the Position 3 bits (the high-order bits), with the result $0 \| 0 = 0$. The full result is then procedure by assembling the bitwise results and arranging them, right to left, in lowest to highest bit order: 0100. If we arrange the problem vertically, you will see that we are simply applying the Boolean operator to each column, as shown below.

Notice that in Micro operations 7 and 8, for the Boolean operators AND and OR, we have changed our usual notation. We normally use the plus sign to indicate OR, and the dot to indicate AND. In micro instructions, we switch to the symbols $\|$ and $\vee$. This is because we cannot use the plus sign as the OR operator, since we are already using it to denote addition. That is, if we see $X \leftarrow X+Y$, an addition operation is being performed. So, to denote the OR operation we need a new notation.

j) The next category of micro operations is that of the shift operations. Shift operations perform the same type of task as is done by a shift register. They move the bits of an operand either to the left, or to the right.

1. Logical Shift left: $X \leftarrow$ shl X
2. Logical Shift right: $X \leftarrow$ shr X
3. Circular Shift left: $X \leftarrow$ cir X

4. Logical Shift right: $X \leftarrow$ cil X
5. Arithmetic Shift left: $X \leftarrow$ ash 1 X
6. Arithmetic Shift right: $X \leftarrow$ ashr X

The logical shift is a shift in which c in = 0. That is to say, the vacated bit is filled with 0. For example, consider the 4-bit value 0110. If we were to shift it left, each bit would move one bit up, and the low-order bit would be filled with 0, resulting in 1100. If we were to shift it right, each bit would move down one bit, and the high-order bit would be filled with 0, resulting in 0011.

The circular shift, also called a rotate, is defined by the rule c in = c out. That is to say, the vacated bit is filled with the bit that falls off the other end. Another way of saying this that the bit that falls off the end of the value wraps around, and comes back in to fill the vacant spot. If we shift the 4-bit value 1010 left, with a circular shift, all bits move up one position. The high-order 1 drops off, and wraps around to fill the low-order bit, resulting in the value 0101. If 1010 is shifted right, all bits move down. The rightmost 0 drops off, and wraps around to fill the high-order slot, resulting in the same value, 0101.

The arithmetic shift must be considered in two separate cases, depending on whether you are shifting left or right. For a left shift c in = 0. You will observe that the arithmetic shift left is the same as the logical shift left. The difference is only in the name. The names of the shifts reflect their usage, rather than any actual operational difference.

The name *arithmetic shift* emphasizes that a shift can be, and is being used to perform arithmetic. Shifting left is an easy way to perform multiplication. For example, in decimal, if we take the number 3, and shift it left one digit, we get 30. This is the result of multiplication. For example, in decimal, if we take the number 3, and shift it left one digit, we get 30. This is the result of multiplying the number 3 by the base of the number system, 10. If we then shift 3 left again, we get 300, which is the result of multiplying 3 by 10, twice.

In general, left shifts can be thought of as multiplying numbers by powers of the base. This extends to binary. For example, if we take the 8-bit number 00000011 (decimal 3) and shift it left, we get 00000110 (decimal 6), which is the result of multiplying the 3 by the base 2. Shifting again gives us 00001100 (decimal 12), which is the result of multiplying by 2 again.

The arithmetic right shift represents division by 2, and is defined by c in = sign. The sign bit is the high-order bit of a binary number. To illustrate the workings of the right shifts, let's look at a couple of examples. First, consider the 8-bit number 00010111 (decimal 23). If we shift this number to the right we get 00001011. For now, we are doing integer division, 11 is, in fact, the result of dividing 23 by 2. (Remember that if we divide 23 by 2, using integer division, the result is $\lfloor 23/2 \rfloor = 11$.

As a second example, consider the 8-bit number 11101001. This number, with sign bit 1, is a negative number (-23 in decimal). If we shift this number right, and use 0 as the carry-in, the result is 01110100 (decimal +116). This result does not represent $\lfloor -23/2 \rfloor = -12$. it is not even negative, as it should be! However, if we use 1 as the carry-in, instead of 0, we get 11110100, which is, in fact, -12 in decimal. What we have learned is that if we desire a right shift that represents division by 2, for negative numbers the

carry-in must be 1, and for non-negative numbers the carry-in must be 0. And equivalent, but more succinct rule is c in = sign, as previously stated. That is to say, when shifting right, all bits are moved down. This includes the high-order bit, which is moved down one bit, but also copied back into the vacant spot at the left.

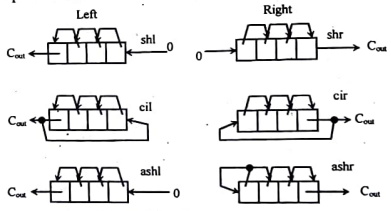


Fig: Illustration of the three shift types

In figure above we summarize the three types of shifts. The logical shift shows a carry-in of 0. The circular shift shows the carry-out routed to the carry-in. The arithmetic shift shows a carry-in of 0 for the left shift, and the sign bit as the carry-in for the right shift.

Unit 2: Micro Programmed Control

Unit at a glance:

2.1 Micro Programmed Control

This is a control unit whose binary variable (i.e. the control functions that specifies a microoperation) remains stored in memory i.e. such type of control unit is software (i.e., microinstruction) based.

Control Words: Control words are words whose bits are used to control certain specified microoperations or general operations. These are basically string of 1's and 0's. Each of the bits in different control words generates different microoperations related to the instruction. Control words are generally stored within control memory.

Routine: Meaningful sequence of instructions is called a routine (or a program).

Microroutine: Microinstructions are stored in control memory in groups. Each of these groups specifies a **microroutine.** So a meaningful sequence of microinstructions constitutes a microroutine. Now the microinstructions within a microroutine must be sequenced and there must be options of branching from one routine to another.

Microinstructions: Microinstruction is an instruction whose bits carry out a set of microoperations at the same time.

Microprogram: A meaningful sequence of microinstructions constitutes a **microprogram.**

Basic Idea Behind the Microprogrammed Control Unit: In this type of control unit, the control signals are generated by means of software. Microprogrammed control unit is microinstructions based.

A microprogrammed control unit will have the following parts:

1) **Control Memory Address Register (CMAR):**

Just like the main memory address register (MAR), inside the control unit there is the CMAR. It specifies the address of the microinstruction to be executed. CMAR holds, in the control memory, the starting address of the microinstruction, to be executed.

2) **Next Address Generator (sequencer):**

It is sometimes also called the microprogram sequencer. It determines the address sequence of the microinstructions that is read from control memory i.e. in the opcode portion of the Instruction Register (i.e. opcode of IR); the opcode of the instruction fetched from the main memory is kept. The opcode is then decoded and thus the starting address of the first microinstruction in the control memory is found. The starting address than comes to the next address generator. So the generator determines the address sequences, i.e. as soon as one microinstruction is executed, the generator generates the address of the next microinstruction and transfers it to the CMAR.

So, the typical ***functions of the generator*** are:

a) It increments the CMAR by one (i.e. loads one address after another into the CMAR).
b) Loads into the CMAR, the next address of the microinstruction in the control memory to be executed.

An initial address gets loaded in the generator to start the control operations.

3) Control Memory Data Register (CMDR):

This acts just like the main memory data register. CMDR holds the present microinstruction read from the control memory while the next address is computed meanwhile and read from control memory.

The data register is sometimes called a pipeline register. This is because, at a time, it allows the execution of the microoperations specified by the control word (i.e. it holds the control word fetched from the control memory and allows it to get executed) and it also allows the generation of the next microinstruction. So, it performs two works simultaneously.

4) Control Memory:

A computer with a microprogrammed control unit has two separate memories: a main memory & a control memory.

The control memory present inside the control unit is assumed to be a ROM which holds fixed microprograms that cannot be altered by occasional users (i.e. while designing a control memory for a particular computer, say an Intel machine the designer knows that in the Intel machine's instruction set, there are 'n' number of instructions. So, for these 'n' instructions, the designer may develop, say 'n^4' number of microinstructions or 'n' number of microprograms and stored them inside the control memory. Now while executing an instruction, only the microinstructions meant for that particular instruction gets executed. So everything in the control memory is pre-developed & they cannot be altered occasionally).

The microprogram, consists of microinstructions that specify various internal control signals for execution of register microoperations i.e. each bit of the microinstruction fetched from the control memory performs some specific logic microoperations.

Working of a Microprogrammed Control Unit:

Steps:

a) Instruction is fetched from the main memory and is stored in the IR.
b) Opcode portion of the IR, which holds the operation part of the instruction, is decoded with a decoder.
c) Decoding the opcode, gives the 1st address or the starting address of the microinstructions in the control memory.
d) The starting address then comes to the 'next address generator'.
e) From there, it goes to the CMAR.
f) Then the control memory is accessed and 1st microinstruction from the starting address location in the control memory is sent to CMDR.

g) The CMDR now holds the 1st microinstruction. Simultaneously, the CMDR asks for the next address information to the next address generator.
h) While asking for the next address information, the CMDR executes the present control word stored in it.
i) On output of the microinstruction, the respective required control signal is generated.
j) Meanwhile, the next address generator on getting 'next-address information' signal from the CMDR, generates the next location address of the next microinstruction in the control memory & sent it to the CMAR.
k) This cycle continues till the execution of the current microprogram (i.e., execution of one instruction) is over.
l) Once execution of one instruction is over, execution starts for the next instruction.

2.2 Concept of Horizontal and vertical microprogramming

The format of the control part of a microinstruction allows us to classify microinstructions into horizontal and vertical.

(a) Horizontal Microinstruction Format:

In a horizontal microinstruction format, each bit in the control field of the microinstruction is attached to a specific control line. Depending on the number of internal processors, control lines and system bus control lines, are connected to single bits. Execution of horizontal microinstruction needs to activate all control lines, which are high leaving apart those, which are low. So the control signals that are activated in result (for control lines which are high) will execute one or more respective microoperations. Length of horizontal microinstructions ranges between 40 to 100 bits. Horizontal microinstructions control many resources, which operate in parallel. A single horizontal microinstruction might control the simultaneous and independent operation of one or more ALUs, input and output to main memory, conditional next address generation, etc.

Advantage: Executions of such microinstructions are fast and need less complicated logic circuitry.

Disadvantage: They need more bits i.e. they less compact.

(b) Vertical Microinstruction Format:

Unlike horizontal microinstruction, in a vertical microinstruction each microoperation to get performed needs specific codes (e.g MAR ← [R1], MAR← PC etc.) (as has been shown in the diagram). Such codes get translated into individual control signals by the decoder in the control unit. Length of vertical microinstructions is short and ranges between 16 to 40 bits. They're called vertical because they are normally listed vertically on a page. Vertical microinstructions effect single operations such as load, add, store, branch, etc. They often resemble machine language instructions containing one opcode and two operands.

Advantage: Such instructions need much less bits compared to horizontal microinstructions i.e. they are more compact.

Disadvantage: Execution of such microinstructions is more time consuming (i.e. delay is more) and needs complex hardware circuitry.

2.3 Comparison between hardwired Control unit and microprogrammed control unit

Hardwired systems are made to perform in a set manner, implemented with logic, switches, etc. between any input and output in the system. Once the manner in which the control is executed, you cannot change the behavior of the system.

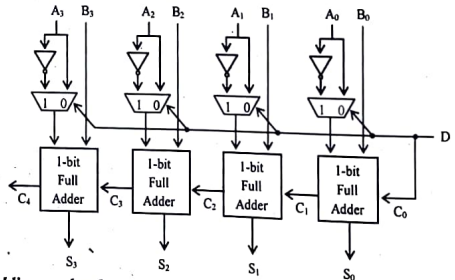
Microprogrammed systems are centered around a computer of some sort, often a microcontroller in small systems, that controls the system using a program. Input is sent to the computer, and the program determines what should be done with the input to come up with an output. So the processor is between the input and the output, rather than a direct link between the input and output.

The vesatility of the microprogrammed system far exceeds the hardwired system. The systems can also be considerably smaller. The size of a complex microcontroller can be quite a bit smaller that a bunch of logic and switches for the same functionality.

2.4 Addition/Subtraction unit block diagram and function

Addition and subtraction are similar algorithms. Taking a look at subtraction, we can see that: $a-b=a+(-b)$

Using this simple relationship, we can see that addition and subtraction can be performed using the same hardware. Using this setup, however, care must be taken to invert the value of the second operand if we are performing subtraction. Note also that in twos-compliment arithmetic, the value of the second operand must not only be inverted, but 1 must be added to it. For this reason, when performing subtraction, the carry input into the LSB should be a 1 and not a zero.



Steps for adding and subtracting numbers in signed 2's complement representation with the help of flowcharts

Steps for addition

Step 1: The augend is put in AC and the addend is put in BR.

Step 2: If any of the number is negative, then its 2's complement representation must be considered.

Step 3: The sum is obtained by adding the contents of AC and BR (including their sign-bits) in the parallel adder and complementer block.

Step 4: The result is stored in AC. If the XOR of the last two carries is 1, then there is a carryout and the overflow bit V is set to 1 (else it is cleared to 0).

Step 5: If there is an overflow then the carryout bit must be discarded. If the result is negative then its 2's complement must be considered to get the actual positive number and then its negative integer representation is considered as the ultimate result.

Steps for subtraction

Step 1: The minuend is put in AC and the subtrahend is put in BR.

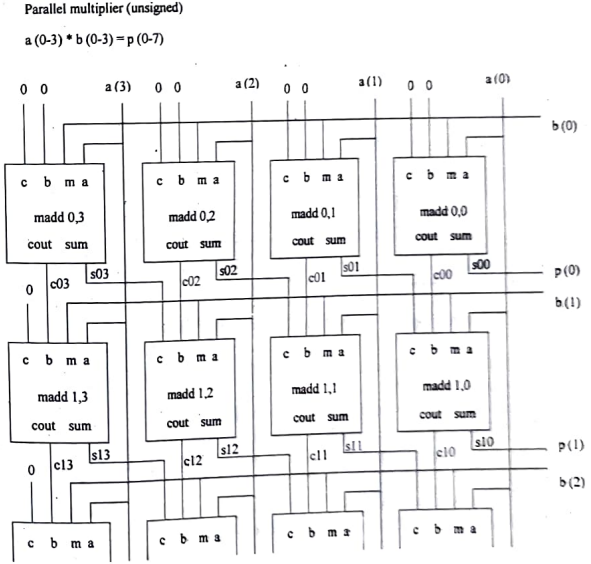
Step 2: If any of the number is negative, then its 2's complement representation must be considered. Also the subtraction operation is accomplished by adding the content of AC to the 2's complement of B. This is because, taking the 2's complement of B, has the effect of changing a negative number to positive and a positive number to negative.

Step 3: The result is obtained by performing the computation in the parallel adder and complementer block.

Step 4: The result is stored in AC. An overflow is checked similarly.

Step 5: The carryout bit must be discarded provided there is an overflow.

2.5 Multiplication circuit diagram and multiplication of positive numbers



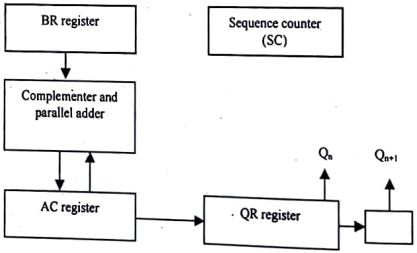
Close observation of the example above reveals that binary multiplication can be performed by scanning the bits of the multiplier right to left, testing each bit, adding or not adding the multiplicand to the product depending on whether the bit is 1 or 0, then left shifting the multiplicand after each "test". Hence the following "test and shift" algorithm for binary multiplication. In the c-code program segment presented below, the testing of each bit of the multiplier is done by right shifting the multiplier and testing the right most bit.

```
p = 0; // initialize product p to 0
while (n != 0) // while multiplier n is not 0
{
if ((n & 0x01) != 0) // test lsb of multiplier
p = p + m; // if 1 then add multiplicand m
m = m << 1; // left shift multiplicand
n = n >> 1; // right shift multiplier
```

2.6 Multiplication of negative numbers and Booths algorithm and its flowchart with example

Booth's algorithm provides a procedure by which binary integers in signed-2's complement representation (i.e. multipliers can be positive or negative) can be multiplied.

Architectural diagram



Explanation of Booth's algorithm with the help of a flowchart

Steps

(a) Initially there is no partial product and hence AC is cleared to 0.

(b) Q_{n+1} is initially cleared to 0.

(c) SC initially holds the number of bits 'n' in the multiplier.

(d) If the two bits Q_n and $Q_{n+1} = 10 \Rightarrow$ first 1 in a string of 1's has been encountered and hence the multiplicand is to be subtracted from the partial product value in AC register.

(e) If the two bits Q_n and $Q_{n+1} = 01 \Rightarrow$ first 0 in a string of 0's has been encountered and hence the multiplicand is to be added to the partial product value in AC register.

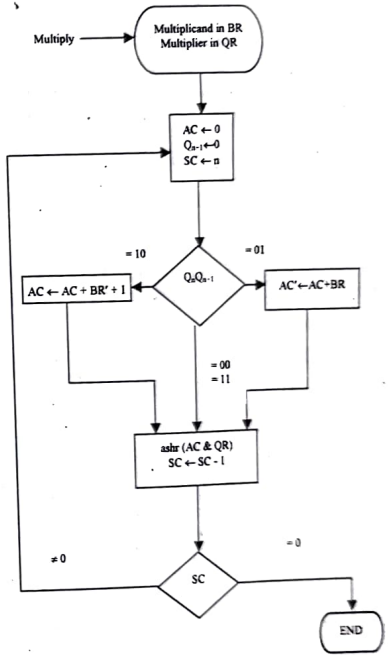
(f) If the two bits Q_n and $Q_{n+1} = 00$ or 11 (i.e. they are equal) $\Rightarrow$ the partial product value remains unchanged. In this technique, overflow cannot occur as the two numbers that are added always have opposite signs.

(g) The partial product (i.e. AC) and the multiplier (i.e. QR) are shifted right. However, the sign-bit in AC remains unchanged.

(h) If SC = 0, the process stops else it continues.

Flowchart

The flowchart for Booth's algorithm is shown below.



2.7 Restoring and non-restoring division process with flowchart and example

Restoring division

Restoring division operates on fixed-point fractional numbers and depends on the following assumptions:

- $D < N$
- $0 < N, D < 1$.

The quotient digits q are formed from the digit set $\{0,1\}$.

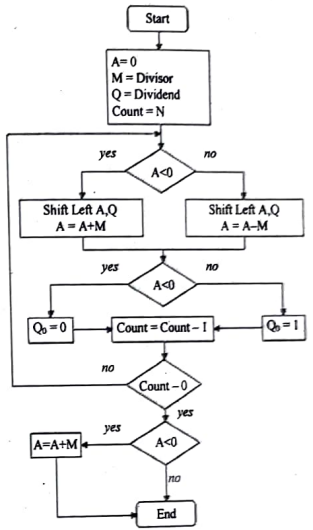
Non-restoring division

Non-restoring division uses the digit set $\{-1, 1\}$ for the quotient digits instead of $\{0, 1\}$.

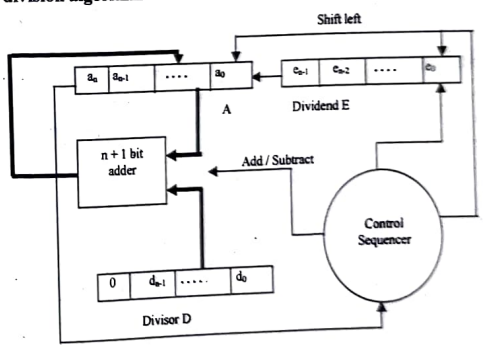
Restoring division technique is the hardware method of performing division operations. Here after each division step, the partial remainder obtained, restored by adding the divisor to the negative difference. This is done to get back the original AC value or to restore the value after every division step.

Non-restoring division technique, if the difference is negative then the divisor is not added directly to the partial remainder. It is added only after shifting the negative difference to the left i.e. suppose while performing division by restoring division technique, subtraction of the divisor content in D from that of A leads to a negative result (i.e. an unsuccessful subtraction). Still the value of A is to be restored. This is however time consuming and can be seen as an unnecessary overhead. This drawback of the restoring division technique can b avoided in the non-restoring division technique.

Non-restoring division algorithm



Restoring division algorithm



2.8 Floating point addition/subtraction algorithm and flowchart (no example)

Basic Operations

- Addition: X + Y = (Mx * 2Ex-Ey + My) * 2Ey, Ex <= Ey
- Subtraction: X -Y = (Mx * 2Ex-Ey - My) * 2Ey, Ex <= Ey

Procedures for addition/subtraction:

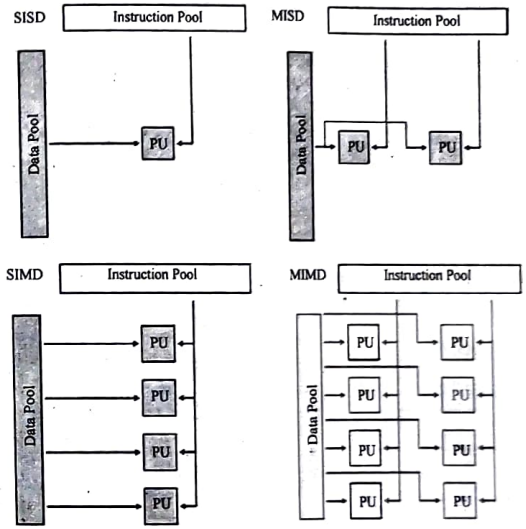
- Adjust exponents and align mantissa
- The exponent of the operands must be made equal for addition and subtraction.
- If Ey>Ex Right shift Mx to form Mx*2Ex-Ey
- If Ex>Ey Right shift My to form My*2Ey-Ex
- Add or subtract mantissa
- Normalize the result
  - Left shift result, decrement result exponent (e.g., if result is 0.001xx...) or
  - Right shift result, increment result exponent (e.g., if result is 10.1xx...)
- Check result
  - Overflow/underflow
  - If result mantissa is 0, may need to set the exponent to zero to return a zero.

2.9 Characteristic features of RISC architecture & Comparison between RISC and CISC

	CISC	RISC
i)	Emphasis on hardware	Emphasis on software
ii)	Includes multi-clock	Single-clock,
iii)	complex instructions	Reduced instruction only
iv)	Memory-to-memory:	Register to register:
v)	"LOAD" and "STORE" incorporated in instructions	"LOAD" and "STORE" are independent instructions
vi)	Small code sizes,	Large code size
vii)	high cycles per second	Low cycles per second
vii)	Transistors used for storing complex instructions	Spends more transistors on memory registers

2.10 Concept of parallel processing and Flynn's Classification

Single Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD)
Multiple Instruction Single Data (MISD), Multiple Instruction Multiple Data (MIMD)
Processing Unit (PU)



2.11 Concept of instruction pipelining

An instruction pipeline is a technique used in the design of computers to increase their instruction throughput (the number of instructions that can be executed in a unit of time). The basic instruction cycle is broken up into a series called a pipeline. Rather than processing each instruction sequentially (one at a time, finishing one instruction before starting the next), each instruction is split up into a sequence of steps so different steps can be executed concurrently (by different circuitry) and in parallel (at the same time).

2.12 Speed-up due to pipelining

The ideal speedup from a pipeline is equal to the number of stages in the pipeline.

$$\frac{\text{Time per instruction on unpiplined machine}}{\text{Number of pipe stages}}$$

2.13 Running the pipeline with minimum idling

The Pipeline pattern uses parallel tasks and concurrent queues to process a sequence of input values. Each task implements a stage of the pipeline, and the queues act as buffers

that allow the stages of the pipeline to execute concurrently, even though the values are processed in order. You can think of software pipelines as analogous to assembly lines in a factory, where each item in the assembly line is constructed in stages. The partially assembled item is passed from one assembly stage to another. The outputs of the assembly line occur in the same order as that of the inputs.

2.14 RISC architecture and pipelining

Every instruction must got through the same stages. Instruction may not require all the stages, but must execute the all, with empty clock cycles if needed.

	IF	ID	EX	MEM	WB	
	Instruction Fetch	Instruction Decode	Execution	Memory Access	Write Back	

Register-type Instructions	IF	ID	EX		WB
Store Instruction	IF	ID	EX	MEM	
Load Instruction	IF	ID	EX	MEM	WB
Branch & Jump Instructions	IF	ID	EX		

Hardware Resource Usage

IF	ID	EX	MEM	WB		
	IF	ID	EX	MEM	WB	
		IF	ID	EX	MEM	WB

time (clock cycles) 1st instrn end 2nd instrn end 3rd instrn end

Independent resources in all stages of execution

- IF – writing IR, Incrementing PC,
- ID – register decoding
- EX – operation with ALU
- MEM – reading or writing the memory
- WB – writing final result to register file
  - ❖ At any moment all resources are being used
  - ❖ At any moment several instructions are processed

2.15 Different pipeline hazards and their detection and minimization

- **Data hazards** arise because of the unavailability of an operand
  - o For example, an instruction may require an operand that will be the result of a preceding, still uncompleted instruction.
- **Structural hazards** may arise from some combinations of instructions that cannot be accommodated because of resource conflicts
  - o For example, if processor has only one register file write port and two instructions want to write in the register file at the same time.
- **Control hazards** arise from branch, jump, and other control flow instructions

  - o For example, a taken branch interrupts the flow of instructions into the pipeline, the branch target must be fetched before the pipeline can resume execution.
- **Common solution is to stall the pipeline until the hazard is resolved, inserting one or more "bubbles" in the pipeline.**

Structural Hazard Solution

❖ **Hardware duplication**

E.g. independent program (IF) and data (MEM) memory block (Harvard Architecture)

❖ **Pipeline stall**

Simplest and some cases necessary, but always brings performance drop

Instr-1	IF	ID	EX	MEM	WB				
Instr-2		IF	ID	EX		WB			
Instr-3			IF	ID	EX	MEM	WB		
Instr-4				stall	IF	ID	EX	MEM	WB

<-2cycles->

Data Hazard Solution

Forwarding

Using up-to-date results directly from earlier execution stages, without waiting for final register update

ADD D1, D2, D3	IF	ID	EX		WB(D3)		
SUB D4, D3, D5		IF	ID	EX		WB	
MUL D4, D3, D1			IF	ID	EX		WB

IF	ID	EX		WB		
	IF	ID	EX		WB	
		IF	ID	EX		WB

Code Optimization

Data hazards can be effectively eliminated by code optimization (by optimizing compilers)

Architecture dependent or independent optimization

e.g. swapping two element of a table:

X[k] i X[k+1]

Register R3 contains address of X[k] table element

Before	After
lw R1,0(R3) * R1 = X[k]	lw R1,0(R3) * R1 = X[k]
lw R2,4(R3) * R2 = X[k+1]	lw R2,4(R3) * R2 = X[k+1]
sw R2,0(R3) * X[k] = R2	sw R1,4(R3) * X[k+1]= R1
sw R1,4(R3) * X[k+1]= R1	sw R2,0(R3) * X[k] = R2

To avoid control hazards

Micro architectures can:

❖ insert a pipeline bubble (discussed above), guaranteed to increase latency, or

❖ use branch prediction and essentially make educated guesses about which instructions to insert, in which case a pipeline bubble will only be needed in the case of an incorrect prediction

2.16 Concept of vector processing, Techniques used in vector processing

Vector Processor (computer)

Ability to process vectors and related data structures such as matrices and multi-dimensional arrays, much faster than conventional computers. Vector Processors may also be pipelined.

Vector Processing Applications

Problems that can be efficiently formulated in terms of vectors
– Long-range weather forecasting
– Petroleum explorations
– Seismic data analysis
– Medical diagnosis
– Aerodynamics and space flight simulations

2.17 Speed advantage of vector processing, Vector processing instruction format

There are two primary types of vector operations:

- Vector-register operations
- Memory-memory vector operations

In vector-register operations, all vector operations—except load and store—are among the vector registers. All major vector computers use vector-register architecture, including the Cray Research processors (Cray-1, Cray-2). In memory-memory vector operations, all vector operations are memory to memory. The first vector computers were of this type, as were CDC's vector computers.

The vector instructions of the following types:

- Vector-vector instructions:
  f1: Vi --> Vj (e.g. MOVE Va, Vb)
  f2: Vj x Vk --> Vi (e.g. ADD Va, Vb, Vc)
- Vector-scalar instructions:
  f3: s x Vi --> Vj (e.g. ADD R1, Va, Vb)
- Vector-memory instructions:
  f4: M --> V (e.g. Vector Load)
  f5: V --> M (e.g. Vector Store)
- Vector reduction instructions:
  f6: V --> s (e.g. ADD V, s)
  f7: Vi x Vj --> s (e.g. DOT Va, Vb, s)
- Gather and Scatter instructions:
  f8: M x Va --> Vb (e.g. gather)
  f9: Va x Vb --> M (e.g. scatter)

- Masking instructions:
  fa: Va x Vm --> Vb (e.g. MMOVE V1, V2, V3)

Gather and scatter are used to process sparse matrices/vectors. The gather operation uses a base address and a set of indices to access from memory "few" of the elements of a large vector into one of the vector registers. The scatter operation does the opposite. The masking operations allow conditional execution of an instruction based on a "masking" register.

2.18 Concept of array processor

Array processor is a computer/processor that has an architecture especially designed for processing arrays (e.g. matrices) of numbers. The architecture includes a number of processors (say 64 by 64) working simultaneously, each handling one element of the array, so that a single operation can apply to all elements of the array in parallel. To obtain the same effect in a conventional processor, the operation must be applied to each element of the array sequentially, and so consequently much more slowly. An array processor may be built as a self-contained unit attached to a main computer via an I/O port or internal bus; alternatively, it may be a distributed array processor where the processing elements are distributed throughout, and closely linked to, a section of the computer's memory.

Array processors are very powerful tools for handling problems with a high degree of parallelism. They do however demand a modified approach to programming. The conversion of conventional (sequential) programs to serve array processors is not a trivial task, and it is sometimes necessary to select different (parallel) algorithms to suit the parallel approach.

2.19 Different types of array processors

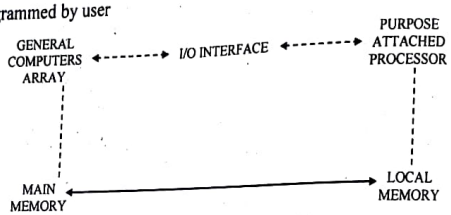
Array processor is of two types:

1) **Attached array processor**
2) **SIMD array processor**

Coming over **attached array processor** - In this an auxillary processor is attached to a general purpose computer. It enhances the performance of the host computer in specific numerical computation tasks.

* It provides vector processing for complex scientific applications.
* This attachment is done by parallel processing i.e. it contains one or more pipelined floating point adders and multipliers.

Can be programmed by user



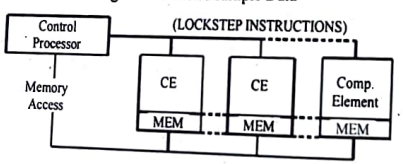
The attached array processor is attached to host computer by a input-output interface or controller and The data for the attached processor are transferred from main memory to a local memory through a high speed bus.

SIMD stand for Single instruction multiple data Instruction.

It consists of multiple processing units or elements that are operating in parallel and are synchronized to perform the same task or operation under the control of a common control unit. HENCE the name is single instruction multiple data.

SIMD

Single Instruction Multiple Data



- Sometimes configured as shared memory & sometimes as distributed memory

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. Which of the following comment about the IR is true? **[WBSCTE 2019]**
   (a) It is used to count the number of instructions
   (b) It is a cell in ROM
   (c) During execution of the current instruction, its content changes
   (d) Opcode fetched from memory stored in IR

Answer: (d)

2. Which of the following rule(s) regarding the addition of 2 given numbers in 2's complement is correct **[WBSCTE 2011, 2015]**
   (a) Add sign bit and discard carry, if any
   (b) Add sign bit and add carry, if any
   (c) don't Add sign bit and discard carry, if any
   (d) Don't Add sign bit and add carry, if any

Answer: (a)

3. When signed numbers are used in binary arithmetic, then which one of the following notations would have unique representation of zero **[WBSCTE 2012]**
   (a) magnitude (b) 1's complement (c) 2's complement (d) none

Answer: (a)

4. Exponent in floating point number can be represented by **[WBSCTE 2013, 2015]**
   (a) Sign–magnitude form, (b) 1's complemented from,
   (c) 2's complemented from, (d) biased form.

Answer: (a)

5. If negative numbers are stored in 1's complemented form, the range of numbers that can be stored in 16 bits is – **[WBSCTE 2013]**
   (a) –128 to +128 (b) –128 to +127
   (c) –127 to +127 (d) –32767 to +32767

Answer: (d)

6. Pipeline architecture is more suitable for- **[WBSCTE 2015]**
   (a) RISC architecture (b) CISC architecture
   (c) Hybrid architecture (d) All of the above.

Answer: (a)

7. Which of the following has least number of instructions? **[WBSCTE 2016]**
   (a) RISC (b) CISC (c) Both a and b (d) XISC

Answer: (b)

8. 01110000 represents **[WBSCTE 2017, 2021]**
   (a) 0 (b) NaN (c) $+\infty$ (d) $-\infty$

Answer: (a)

9. The largest floating point number that can be represented by 8 bit is **[WBSCTE 2017, 2021]**
   (a) 01111111 (b) 11111111 (c) 01101111 (d) 01111110

Answer: (a)

10. If n is number of bits in exponent, the bias number can be calculated as **[WBSCTE 2017, 2021]**

(a) $2^n - 1$ (b) 2^n (c) 2^{n-1} (d) $2^{n-1} - 1$

Answer: (a)

11. In Booth's algorithm, if the multiplier has n bits then the multiplicand should have **[WBSCTE 2017, 2018, 2021]**

(a) 1 bit (b) n bits (c) n+1 bits (d) 2n bits

Answer: (c)

12. 01110010 represents **[WBSCTE 2018]**

(a) 0 (b) NaN (c) $+\infty$ (d) $-\infty$

Answer: (a)

13. If we want an addition/subtraction circuit to do subtraction, the initial value of carry in should be **[WBSCTE 2018]**

(a) –1 (b) 0 (c) 1 (d) 2

Answer: (c)

14. The final addition sum of the numbers, 0.110 & 0110 is _____. **[WBSCTE 2022]**

(a) 1101 (b) 1111 (c) 1001 (d) 1010

Answer: (a)

15. Individual control word of the micro routine are called as **[WBSCTE 2022]**

(a) Micro task (b) Micro instruction
(c) Micro operation (d) Micro Command

Answer: (b)

16. The situation wherein the data of operands are not available is called _____ **[WBSCTE 2022]**

(a) Data hazard (b) Stock
(c) Deadlock (d) Structural hazard

Answer: (a)

17. What is the full form of CISC? **[WBSCTE 2022]**

(a) Complex Instruction Sequential Compilation
(b) Complete Instruction Sequential Compilation
(c) Computer Integrated Sequential Compiler
(d) Complex Instruction Set Computer

Answer: (d)

18. In a Micro-processor, the address of the next instruction to be executed, is stored in **[Model Question]**

(a) Stack pointer (b) Address hatch
(c) Program counter (d) General purpose register

Answer: (c)

19. In microprogramming **[Model Question]**

(a) horizontal microinstruction is faster (b) vertical microinstruction is faster
(c) hardware control unit is faster (d) none of these

Answer: (c)

20. Which logic gate has the highest speed? **[Model Question]**

(a) DTL (b) RTL (c) ECL (d) TTL

Answer: (c)

21. A UART is an example of **[Model Question]**

(a) serial asynchronous data transmission ship
(b) PIO
(c) DMA controller
(d) none of these

Answer: (a)

22. Overlapped register windows are used to speed-up procedure call and return in **[Model Question]**

(a) RISC architectures (b) CISC architectures
(c) both (a) and (b) (d) none of these

Answer: (a)

23. The number of cycles required to complete n tasks in a k stage pipeline is **[Model Question]**

(a) $k + n - 1$ (b) $nk + 1$ (c) k (d) none of these

Answer: (a)

24. A 4-ary 3-cube hypercube architecture has **[Model Question]**

(a) 3 dimensions with 4 nodes along each dimension
(b) 4 dimensions what 3 nodes along each dimension
(c) both (a) and (b)
(d) none of these

Answer: (a)

25. Which of these are examples of 2-dimensional topologies in static networks? **[Model Question]**

(a) Mesh (b) 3CCC networks
(c) Linear array (d) None of these

Answer: (a)

26. The seek time of a disk is 30 ms. It rotates at the rate of 30 rotations/second. The capacity of each track is 300 words. The access time is approximately **[Model Question]**

(a) 62 ms (b) 60 ms (c) 47 ms (d) none of these

Answer: (d)

27. The performance of a pipelined processor suffers if **[Model Question]**

(a) the pipeline stages have different delays
(b) Consecutive instructions are dependent on each other
(c) the pipeline stages share hardware resources
(d) all of these

Answer: (d)

28. A single bus structure is primarily found in **[Model Question]**

(a) Main frames (b) High performance machines
(c) Mini and Micro-computers (d) Supercomputers

Answer: (c)

29. What is a main advantage of classical vector systems (VS) compared with RISC based systems (RS)? **[Model Question]**

(a) VS have significantly higher memory bandwidth than RS
(b) VS have higher clock rate than RS
(c) VS are more parallel than RS
(d) None of these

Answer: (a)

30. The division of stages of a pipeline into sub-stages is the basis for **[Model Question]**

(a) pipelining (b) super-pipelining
(c) superscalar (d) VLIW processor

Answer: (a)

31. Difference between RISC and CISC is **[Model Question]**

(a) RISC is more complex (b) CISC is more effective
(c) RISC is better optimizable (d) none of these

Answer: (a)

32. The advantage of RISC over CISC is that **[Model Question]**

(a) RISC can achieve pipeline segments, requiring just one clock cycle
(b) CISC uses many segments in its pipeline with the longest segment requiring two or more clock cycle
(c) both (a) & (b)
(d) none of these

Answer: (d)

33. Which of the following architectures correspond to von-Neumann architecture? **[Model Question]**

(a) MISD (b) MIMD (c) SISD (d) SIMD

Answer: (c)

34. Utilization pattern of successive stages of a synchronous pipeline can be specified by **[Model Question]**

(a) Truth table (b) Excitation table
(c) Reservation table (d) Periodic table

Answer: (c)

35. SPARC stands for **[Model Question]**

(a) Scalable Processor Architecture
(b) Superscalar Processor A RISC Computer
(c) Scalable Processor A RISC Computer
(d) Scalable Pipeline Architecture

Answer: (a)

36. Portability is definitely an issue for which of the following architectures? **[Model Question]**

(a) VLIW processor (b) Super Scalar processor
(c) Super pipelined (d) none of these

Answer: (b)

B. Fill in the blanks in the following statements:

37. The bias value for single –precision floating point numbers is *127.* **[WBSCTE 2022]**

C. Answer the following questions:

38. What is Opcode? **[WBSCTE 2018]**

Answer:

An opcode is the first byte of an instruction in machine language which tells the hardware what operation needs to be performed with this instruction. Every processor/controller has its own set of opcodes defined in its architecture. An opcode is followed by data like address, values etc if needed.

39. Discuss binary number system. **[WBSCTE 2013]**

Answer:

In mathematics and digital electronics, a binary number is a number expressed in the binary numeral system, or base-2 numeral system, which represents numeric values using two different symbols: typically 0 (zero) and 1 (one). More specifically, the usual base-2 system is a positional notation with a radix of 2. Because of its straightforward implementation in digital electronic circuitry using logic gates, the binary system is used

internally by almost all modern computers and computer-based devices such as mobile phones.

40. What is Program Counter? [WBSCTE 2017, 2018, 2021]

Answer:

A program counter is a register in a computer processor that contains the address (location) of the instruction being executed at the current time. As each instruction gets fetched, the program counter increases its stored value by 1. After each instruction is fetched, the program counter points to the next instruction in the sequence. When the computer restarts or is reset, the program counter normally reverts to 0.

41. Write full form of RISC and CISC. [WBSCTE 2017, 2021]

Answer:

Full form of RISC and CISC --- Reduced instruction set computing and Complex instruction set computing.

42. What is the minimization of pipeline? [WBSCTE 2018]

Answer:

The minimization of pipeline refers to hazard minimization in pipeline. It can be done by:

1. Data hazards can be effectively minimized by code optimization (by optimizing compilers).
2. To avoid control hazards microarchitectures can insert a pipeline bubble (discussed above), guaranteed to increase latency, or use branch prediction and essentially make educated guesses about which instructions to insert, in which case a pipeline bubble will only be needed in the case of an incorrect prediction

43. How control unit controls other units? [WBSCTE 2022]

Answer: The control unit controls other units by generating control signal.

44. What are the three main elements of the control unit? [WBSCTE 2022]

Answer:

The components of the Hardwired control unit are instruction register (contains opcode and address field), timing unit, control state generator, control signal generation matrix, and instruction decoder.

45. What is control memory address?

Answer: [WBSCTE 2022]

The control memory address register specifies the address of the microinstruction, and the control data register holds the microinstruction read from memory.

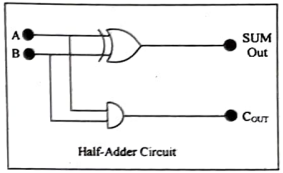
46. What is the 2's complement representation of –6? [WBSCTE 2022]

Answer:

0000 0110

47. Draw the block diagram of the half adder. [WBSCTE 2022]

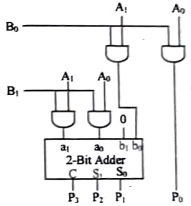
Answer:



Half-Adder Circuit

48. Draw a multiplication circuit diagram. [WBSCTE 2022]

Answer:



Schematic of 2×2 Bit Multiplier Using 2-Bit Full Adder

49. What is RISC Pipeline? [WBSCTE 2022]

Answer:

RISC stands for Reduced Instruction Set Computers. It was introduced to execute as fast as one instruction per clock cycle.

50. What size of MUXs are needed? [WBSCTE 2022]

Answer: We need a 2^N input multiplexer for N inputs.

51. What do you mean by pipelined chaining? [Model Question]

Answer:

Pipeline chaining is a linking process that occurs when results obtained from one pipeline unit are directly fed into the operand registers of another functional pipe.

52. Why is MIPS rating specified as performance index in computing machines? [Model Question]

Answer:

In the context of CPU performance measurement, MIPS stand for 'Million Instructions Per Second', The MIPS rating of a CPU refers to how many low-level machine code instructions a processor can execute in one second.

53. What is understood by dynamic pipeline? [Model Question]

Answer:

It is a pipeline with dynamic scheduling. Dynamic scheduling is one method for improving performance in a multiple instruction issue processor. When applied to a super scalar processor, dynamic scheduling will boost up performance in the face of data hazards and also allows the processor to potentially overcome the issue restrictions. Dynamic pipeline must be multifunctional. In the dynamic pipeline different reservation table are used for different function.

54. What is the function of reservation table in pipeline architecture system? [Model Question]

Answer:

The functions of reservation table are as follows:

- Reservation table displays the time-space flow of data through the pipeline for one function evaluation.
- Different functions may follow different paths.
- The same table may represent a number of pipeline configurations.
- The number of columns in a reservation table is called the evaluation time.

55. What is parallel algorithm? [Model Question]

Answer:

In computer science, a parallel algorithm, as opposed to a traditional serial algorithm, is an algorithm which can be executed a piece at a time on many different processing devices, and then combined together again at the end to get the correct result.

Many parallel algorithms are executed concurrently – though in general concurrent algorithms are a distinct concept – and thus these concepts are often conflated, with which aspect of an algorithm is parallel and which is concurrent not being clearly distinguished.

56. Compare between RISC and CISC. [Model Question]

Answer:

Characteristics	CISC	RISC
Instruction set size and instruction formats	Instruction set is very large and instruction format is variable (16 – 64 bit per instruction)	Instruction set is small and instruction format is fixed.
Addressing mode	12 - 24	3- 5
General purpose registers and cache design	8-24 general purpose registers present. Unified cache is used for instruction and data	Though most instructions are register base so large numbers of registers (32 – 192) are used and cache is split in data cache and instruction cache.
CPI	CPI is between 2 to 15	In most cases CPI is 1 but average CPI is less than 1.5
CPU control	CPU is controlled by control memory (ROM) using microprograms.	CPU is controlled by hardware without control memory

57. Based on the design characteristics, classify microinstructions. [Model Question]

Answer:

Microinstructions can be classified in the following ways based on the different design characteristics:

a. Vertical and Horizontal
b. packed and Unpacked
c. Hard and Soft
d. Direct and Indirect encoding

58. What do you mean by hard microprograms and soft microprograms? [Model Question]

Answer:

The terms hard and soft microprograms emphasize different design characteristics for a microprogram.

Microprograms that are generally fixed or un-modifiable are entitled as *hard microprograms*. Such microprograms are more committed to read-only memories.

On the other hand, *soft microprograms* are highly modifiable and are meant for user microprogramming.

59. What can be the applications of hardwired and microprogrammed control unit? [Model Question]

Answer:

Hardwired control units are used in the RISC (Reduced Instruction Set Computer) type computers and Microprogrammed control units are used in CISC (Complex Instruction Set Computer) type computers.

60. What is a firmware? [Model Question]

Answer:

A meaningful sequence of microinstructions is known as a microprogram or firmware, which means that microprogram lies in midway between hardware and software.

61. Give advantages & disadvantages of microporgramming. [Model Question]

Answer:

Advantages:

1. Easy to modify the way instructions works, for example, in order to fix up a problem in the programming.
2. Easy to add new instructions.

Disadvantages:

1. High-level machine instructions are too slow or complex.
2. Too complex logic circuitry.
3. Microprogramming is slower than pipelining concept.

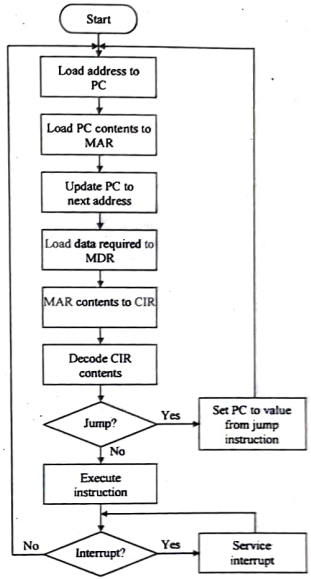
Long Answer Type Questions

➲ **1. Describe the execution sequence of an instruction through different section of CPU.** **[WBSCTE 2013]**

Answer:

The circuits used in the CPU during the cycle are:

- **Program counter (PC)** - an incrementing counter that keeps track of the memory address of the instruction that is to be executed next
- **Memory address register (MAR)** - holds the address of a memory block to be read from or written to
- **Memory data register (MDR)** - a two-way register that holds data fetched from memory (and ready for the CPU to process) or data waiting to be stored in memory
- **Instruction register (IR)** - a temporary holding ground for the instruction that has just been fetched from memory
- **Control unit (CU)** - decodes the program instruction in the IR, selecting machine resources such as a data source register and a particular arithmetic operation, and coordinates activation of those resources
- **Arithmetic logic unit (ALU)** - performs mathematical and logical operations.



The time period during which one instruction is fetched from memory and executed when a computer is given an instruction in machine language. There are typically four stages of an instruction cycle that the CPU carries out:

1) Fetch the instruction from memory.
2) "Decode" the instruction.
3) "Read the effective address" from memory if the instruction has an indirect address.
4) "Execute" the instruction.

➲ **2. (a) Explain the difference between Hardwired Control and Micro-programmed control.**
(b) Explain the organization of control memory in Micro-programmed control with diagram. **[WBSCTE 2014, 2019]**

Answer:

a) ***Refer to Article No. 2.3 of Unit at a glance.***

b) A computer with a microprogrammed control unit has two separate memories: a main memory & a control memory.

The control memory present inside the control unit is assumed to be a ROM which holds fixed microprograms that cannot be altered by occasional users (i.e. while designing a control memory for a particular computer, say an Intel machine the designer knows that in the Intel machine's instruction set, there are 'n' number of instructions. So, for these 'n' instructions, the designer may develop, say 'n^4' number of microinstructions or 'n' number of microprograms and stored them inside the control memory. Now while executing an instruction, only the microinstructions meant for that particular instruction gets executed. So everything in the control memory is pre-developed & they cannot be altered occasionally).

The microprogram, consists of microinstructions that specify various internal control signals for execution of register microoperations i.e. each bit of the microinstruction fetched from the control memory performs some specific logic microoperations.

Say, the CMDR holds the control word or microinstruction, 11001010, from the control memory. Now, on execution, say the first 1 will make the tri-state buffer high, the 2nd 1 will make the control bus high, the 1st 0 will make the chip select line low and so on i.e. on execution, each of the bits in the control word will perform respective microoperation.

If, suppose the microinstructions in a particular machine are 8-bit wide i.e. there are 8-bits in each mcroinstruction. So while execution, the output of the CMDR will have 8 separate lines. Each of the lines will be connected to respective logic circuitry to generate the control signal Depending on the control word, respective individual line will be enabled.

Suppose, the CMDR holds 11001010.

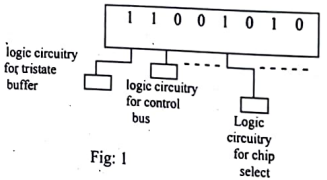


Fig: 1

So, as per figure 1, each of the 8 separate lines (as there are 8-bits in the microinstruction) is connected to different logic circuits. According to the bits in the microinstruction, the respective lines are enabled / disabled and the corresponding microoperations are performed & thus the required control signals are generated.

3. a) Explain the difference between Hardwired Control and Microprogrammed control.

b) What do you mean by horizontal and vertical microprogramming? Compare these two ways of microprogramming. **[WBSCTE 2016, 2017, 2021]**

Answer:

a) To execute an instruction, there are two types of control units Hardwired Control unit and Micro-programmed control unit.

1. Hardwired control units are generally faster than microprogrammed designs. In hardwired control, we saw how all the control signals required inside the CPU can be generated using a state counter and a PLA circuit.
2. A microprogrammed control unit is a relatively simple logic circuit that is capable of (1) sequencing through microinstructions and (2) generating control signals to execute each microinstruction.
3.

Hardwired Control Unit	Micro-programmed Control Unit
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micro-programmed control unit generates the control signals with the help of micro instructions stored in control memory
Hardwired control unit is faster when compared to micro-programmed control unit as the required control signals are generated with the help of hardwares	This is slower than the other as micro instructions are used for generating signals here
Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating control signals
It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions
Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)

b) Vertical and Horizontal Micro-programming:

There are two distinct ways of organising microinstructions. In the case of vertical microprogramming, each line of the micro-program represents a micro-instruction which specifies one or more micro-operations. One micro-instruction gets executed during each step of the control sequence. One can use a straight binary code to specify each micro-operation. Rather than provide for only one micro-operation per step and use a single decoder for the entire micro-instruction field, it is in fact possible to partition this field

into a number of mutually exclusive sub-fields which can be independently decoded in separate decoders. This approach yields a more cost-effective design. If the system design needs in all a total of N different micro-operations, one will have to provide for $\log_2 N$ bits to specify a micro-operation. if only one micro-operation per micro-instruction is allowed, then one would require $\log_2 N$ bits per micro-instruction.

In horizontal micro-programming, one associates each bit of the micro-instruction with a specific micro-operation (bit I to represent micro-operation I). A specific micro-operation is executed during a micro-instruction step only if the corresponding bit is one. For instance, micro-operation I is executed if the Ith bit is one; it is not executed otherwise. A micro-code decoder will no longer be required in this casse because no coding is involved. This scheme will require to total of N bits per micro-instruction (to accommodate N different micro-operations). (It can thus be seen that an obvious distinction between horizontal or vertical micro-programming is the number of bits used per micro-instruction in the control memory). The advantage with horizontal micro-programming is that several micro-operations can be executed during each micro-instruction step. In fact, this scheme permits one to execute all the micro-operations provided in the computer in a single micro-instruction. This, of course, is wasteful as this degree of parallelism would never be needed.

As against the advantage of efficiency offered by vertical micro-programming, in horizontal micro-programming, it is possible to individually and independently control each micro-operation in the computer. There is no restriction on the extent of parallelism between them. It is therefore particularly suitable for implementing high-speed computers.

Vertical and horizontal micro-programming schemes are illustrated in Fig. 1.

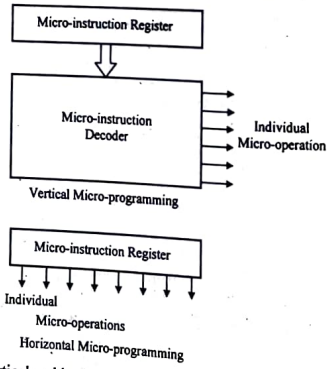


Fig: 1 Vertical and horizontal micro-programming.

➲ 4. Explain how priority of interrupt will be implemented? **[WBSCTE 2019]**

Answer:

When I/O devices are ready for I/O transfer, they generate an interrupt request signal to the computer. The CPU receives this signal, suspends the current instructions it is executing and then moves forward to service that transfer request. But what if multiple devices generate interrupts simultaneously. In that case, we have to have a way to decide which interrupt is to be serviced first. In other words, we have to set a priority among all the devices for systemic interrupt servicing.

The concept of defining the priority among devices so as to know which one is to be serviced first in case of simultaneous requests is called priority interrupt system. This could be done with either software or hardware methods.

SOFTWARE METHOD – POLLING:

In this method, all interrupts are serviced by branching to the same service program. This program then checks with each device if it is the one generating the interrupt. The order of checking is determined by the priority that has to be set. The device having the highest priority is checked first and then devices are checked in descending order of priority. If the device is checked to be generating the interrupt, another service program is called which works specifically for that particular device.

The structure will look something like this-

```
if (device[0].flag)
    device[0].service();
else if (device[1].flag)
    device[1].service();
.else
    //raise error
```

The major disadvantage of this method is that it is quite slow. To overcome this, we can use hardware solution, one of which involves connecting the devices in series. This is called Daisy-chaining method.

HARDWARE METHOD – DAISY CHAINING:

The daisy-chaining method involves connecting all the devices that can request an interrupt in a serial manner. This configuration is governed by the priority of the devices. The device with the highest priority is placed first followed by the second highest priority device and so on.

➲ 5. Explain how addition operation is done with floating point numbers. **[WBSCTE 2007, 2009]**

Answer:

The addition operation is done with algorithm similar to that used on sign magnitude integers.

Let, first number X is represented by and second number is A simple method to add floating-point numbers is to first represent them with the same exponent, which is done

by subtracting them and aligning the mantissas by shifting one of them until the difference the exponents has been reduced to zero. Next the aligned mantissas are added. Finally, the result is normalized, if necessary, by again shifting the mantissa and making a compensating change in the exponent. The mantissa and exponent of the final result are placed in the two registers. Floating point overflow and underflow are also being performed.

For example: and, we first match the exponents of X and Y. Compute X +Y= 1.32+2.6 = 3.92. So, the final result has mantissa 3.92 and exponent 4.

6. Write a brief note on Booth's Algorithm. [WBSCTE 2008, 2010]

OR,

Show how multiplication is done in computer. [WBSCTE 2009]

OR,

Describe Booth's algorithm with suitable architectural diagram. [WBSCTE 2011, 2014, 2015, 2019]

Answer: *Refer to Article No. 2.6 of Unit at a glance.*

7. Perform the operation (20) – (5) by using 2's complement. [WBSCTE 2008]

OR,

With an example show how subtraction can be done with 2's complement method. [WBSCTE 2010]

Answer:

Let us assume given number is represented by 8-bit representation.

20 is given by 00010100 while –5 in its two's complement representation is given as 11111011. On addition of these two numbers the result obtained is ***11101110*** or –18 (the two's complement form of the result is 00010010 which is binary equivalent of + 18. hence –18 is the result).

Converting decimal to binary,

$(20)_{10} = (00010100)_2$

$(5)_{10} = (00000101)_2$

Again 2's complement of $(00000101)_2 = (11111011)_2$

Adding both numbers,

Ignoring the carry bit we get $(00001111)_2 = (15)_{10}$

8. Perform the multiplication (1001) * (1101) by using binary numbers. [WBSCTE 2010]

Answer:

```
  1001
  1101
  1001
 0000x
1001xx
1001xxx
1110101
```


9. Show the steps of multiplication performed by using Booth's algorithm of –7×5. Assume that both numbers are in 4 bit representation. [WBSCTE 2011]

OR,

Show the steps of multiplication performed by using Booth's algorithm of –7×5. [WBSCTE 2014, 2015]

Answer:

Multiplicand -	
Decimal:	-7
Binary:	11111001
Multiplier -	
Decimal:	5
Binary:	00000101
Two's Complement:	11111011
Steps -	
Starting Out:	0000000011111001
Subtract:	1111101111111001
Shift:	1111110111111100
Add:	0000001011111100
Shift:	0000000101111110
Shift:	0000000010111111
Subtract:	1111101110111111
Shift:	1111110111011111
Shift:	1111111011101111
Shift:	1111111101110111
Shift:	1111111110111011
Shift:	1111111111011101
Final Product (Binary):	1111111111011101
Final Product (Decimal):	-35

➲ **10. (a) Write down the flowchart of both restoring and non-restoring division of integer numbers.** [WBSCTE 2011, 2014, 2015, 2019]

(b) Show the non-restoring division steps of 13/3. [WBSCTE 2011, 2014, 2015]

OR,

Write the algorithm of Restoring Division process. Show the restoring division steps of 12/3. [WBSCTE 2017]

Answer:

a) *Refer to Article No. 2.7 of Unit at a glance.*

b) Initially A=00000, M=00011, -M=11101, Q=1101

So, 00000 1101

First Cycle

Shift 00001 101-

Subtract 11101

Set Q0 11110 1010

Second Cycle

Shift 11101 010-

Add 00011

Set Q0 00000 0101

Third Cycle

Shift 00000 101-

Subtract 11101

Set Q0 11101 1010

Fourth Cycle

Shift 11011 010-

Add 00011

Set Q0 11110 0100

Add 00011

00001

Remainder = 00001=1 Quotient =0100 = 4

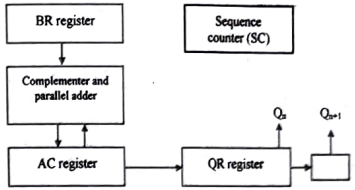
➲ **11. (a) Draw the flowchart of Booth's algorithm for signed multiplication and multiply the following signed 2's complement numbers. Multiplicand = 110011, Multiplier = 101100**

(b) Perform the operation $(-10)_{10} - (-5)_{10}$ by using 8-bits 2's complement form. [WBSCTE 2012]

Answer:

a) 1st Part: Booth's algorithm provides a procedure by which binary integers in signed-2's complement representation (i.e. multipliers can be positive or negative) can be multiplied.

Hardware configuration for Booth's multiplication algorithm Diagram



The figure shows the hardware implementation for Booth's algorithm.

***Register BR*:** It holds the multiplicand along with its sign-bit.

***Register AC*:** It holds the partial product after each stage of multiplication.

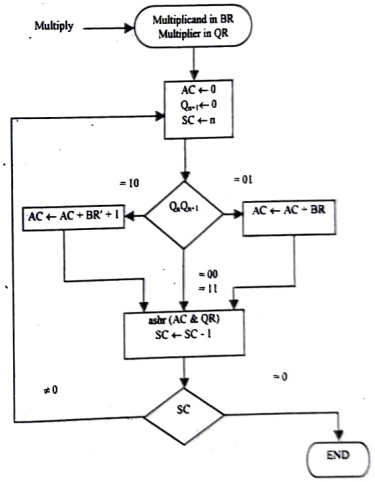
***Register QR*:** It holds the multiplier along with its sign-bit.

***Q_n*:** It designates the least significant bit of the multiplier.

***Q_{n+1}*:** This is a flip-flop with the purpose of double-bit inspection of the multiplier.

Sequence Counter (SC): Keeps track of the number of bits in the multiplier and decrements by 1 after each multiplier bit is multiplied to the multiplicand.

Booth algorithm gives a procedure for multiplying binary integers in signed –2's complement representation.



2nd Part:
Multiplicand : 110011 (–13)
Multiplier : 101100 (–20)

		1	0	1	1	0	0	Multiplicand
		–1	+1	0	–1	0	0	Recoded multiplier

Multiplication:

						1	1	0	0	1	1	Multiplicand
					×	–1	+1	0	–1	0	0	Recoded multiplier
0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0		
0	0	0	0	0	0	1	1	0	1			← 2's complement of the multiplicand
0	0	0	0	0	0	0	0	0				
1	1	1	1	0	0	1	1					
0	0	0	1	1	0	1						← 2's complement of the multiplicand
0	0	0	1	0	0	0	0	0	1	0	0	

The answer is: 260

b) $(10)_{10} \rightarrow 00001010$
$(-10)_{10} \rightarrow 11110110$
$(5)_{10} \rightarrow 00000101$
$(-5)_{10} \rightarrow 11111011$
$(-10)_{10} - (-5)_{10}$
$-10 \rightarrow 11110110$
$-(-5) \rightarrow 00000101$
11111011

The sign bit is 1. Therefore, the answer is negative and is in 2's complement form.
1' complement of the answer is 11111010
The number is 00000101 = –5
Ans. $(-5)_{10}$

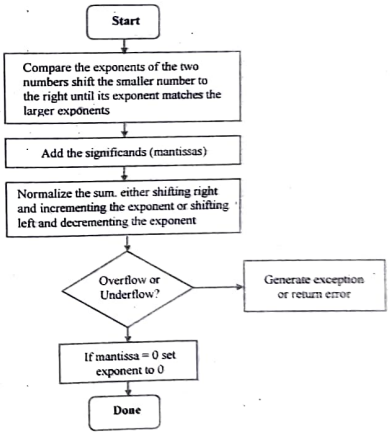
➲ **12. Draw and explain the flow-chart for floating point addition/multiplication operation. What is normalized floating point number?** **[WBSCTE 2012]**

Answer:
1st Part:
Floating point addition:
Assuming that the operands are already in the IEEE 754 format, performing floating point addition: Result = X + Y = $(Xm \times 2^{Xe}) + (Ym \times 2^{Ye})$ involves the following steps:
(1) Align binary point:
- Initial result exponent: the larger of Xe, Ye
- Compute exponent difference: Ye – Xe
- If Ye>Xe Right shift Xm that many positions to form Xm 2^{Xe-Ye}
- If Ye>Xe Right shift Xm that many positions to form Ym 2^{Ye-Ye}

(2) Compute sum of aligned mantissa:
i.e., $Xm2^{Xe-Ye} + Ym$ or $Xm + Xm2^{Ye-Xe}$
(3) If normalization of result is needed, then a normalization step follows:
- Left shift result, decrement result exponent (e.g., if result is 0.001xx…) or
- Right shift result, increment result exponent (e.g., if result is 10.1xx…)

Continue until MSB of data is 1 (note: Hidden bit in IEEE Standard)
(4) Check result exponent:
- If larger than maximum exponent allowed return exponent overflow
- If smaller than maximum exponent allowed return exponent underflow

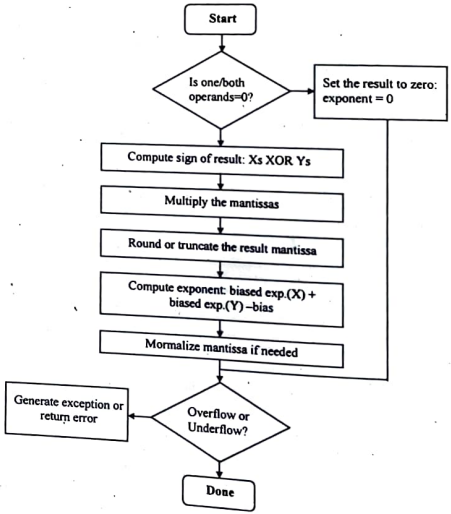
(5) If result mantissa is 0, may need to set the exponent to zero by a special step to return a proper zero.



Floating point multiplication:
Assuming that the operands are already in the IEEE 754 format, performing floating point multiplication:
Result = R = X * Y = $(-1)^{Xs}$ (Xm × 2^{Xe}) * $(-1)^{Ys}$ (Ym × 2^{Ye}) involves the following steps:
(1) If one or both operands is equal to zero, return the result as zero, otherwise:
(2) Compute the sign of the result Xs XOR Ys
(3) Compute the mantissa of the result:
- Multiply the mantissa: Xm * Ym
- Round the result to the allowed number of mantissa bits

(4) Compute the exponent of the result:
Result exponent = biased exponent (X) + biased exponent (Y) – bias
(5) Normalize if needed, by shifting mantissa right, incrementing result exponent.
(6) Check result exponent for overflow/underflow:
- If larger than maximum exponent allowed return exponent overflow

If smaller than minimum exponent allowed return exponent underflow.



2nd Part:
For each floating-point number there is one representation that is said to be *normalized*. A floating-point number is normalized if its mantissa is within the range defined by the following relation:
1/radix <= mantissa < 1
Normalization:
Un-normalized: F = 0.0101 E = 0011 N = 5/16 x 2^3 = 5/2
Normalized: F = 0.101 E = 0010 N = 5/8 x 2^2 = 5/2

13. (a) Draw the flowchart of restoring division method. Apply restoring division procedure to divide decimal no. 7 by decimal no. 3 in 4-bits form.
(b) How can the non-restoring division algorithm be deduced from restoring division algorithm? **[WBSCTE 2012]**

Answer:
a) 1st Part: ***Refer to Article No. 2.7 of Unit at a glance.***
2nd Part:
Q = 7 = 0111 and divisor M = 3 = 0011.

	M	A	Q	Size
Initial Configuration	00011	00000	0111	4
Step-1				
LS(AQ)	00011	00000	111–	—
A=A – M	00011	11101	111–	—
As Sign of A= –ve				
Set Q[0] = 0				
& Restore A	00011	00000	1110	3
Step-2				
LS(AQ)	00011	00001	110–	—
A=A – M	00011	11110	110–	—
Set Q[0] = 0				
Restore A	00011	00001	1100	2

b) Suppose while performing division by restoring division technique, subtraction of the divisor content in D from that of a leads to a negative result (i.e. an unsuccessful subtraction). Still the value of A is to be restored. This is however time consuming and can be seen as an unnecessary overhead. This drawback of the restoring division technique can be avoided in the non-restoring division technique that works in the following manner:
After the subtraction is done in the restoring division technique, provided that sign of A positive (i.e. if A ← 0), then both the contents in A and E are shifted one bit to the left and D is subtracted from A (i.e. 2A – D) else if A is negative (i.e. sign of A ← 1) again both the contents of A and E are shifted left by 1 bit but D is added to A (i.e. effectively 2A + D is performed). Based on the result of the correct operation the e_0 bit is set to either 0 or 1.

14. (a) Describe Booth's algorithm with suitable architectural diagram and flowchart.

(c) Show the steps of multiplication performed by using Booth's algorithm of 6×(–7). [WBSCTE 2013]

OR,

Draw and explain the flow chart for Booth's Algorithm. Show the steps of multiplication performed by using Booth's algorithm of –6×7. [WBSCTE 2017]

Answer:

a) ***Refer to Article No. 2.6 of Unit at a glance.***

b) Find 6 × (–7), with **m** = 6 and **r** = –7, and $x = 4$ and $y = 4$:

- m = 0110, -m = 1010, r = 1001
- A = 0110 0000 0
- S = 1010 0000 0
- P = 0000 1001 0
- Perform the loop four times:
  1. P = 0000 1001 **0**. The last two bits are 10.
     - P = 1010 1001 0. P=P+S
     - P=1101 0100 1 Arithmetic right shift.
  2. P = 1101 0100 1. The last two bits are 01.
     - P = 0011 0100 1. P=P+A
     - P= 0001 1010 0. Arithmetic right shift.
  3. P = 0001 1010 0. The last two bits are 00.
     - P = 0000 1101 0. Arithmetic right shift.
  4. P = 0000 1101 0. The last two bits are 10.
     - P = 1010 1101 0. P=P+S
     - P= 1101 0110 1.Arithmetic right shift.
- The product is 1101 0110, which is –42.

15. Show the non-resting division steps of 11/3. [WBSCTE 2013, 2015]

Answer:

Initially A=00000, M=00011, -M=11101, Q=1011

So, 00000 1011

First Cycle

Shift	00001	011-
Subtract	11101	
Set Q_0	11110	0110

Second Cycle

Shift	11100	110-
Add	00011	
Set Q_0	11111	1100

Third Cycle

Shift	11111	00011
Set Q_0	00010	1001

Fourth Cycle

Shift	00101	001-
Subtract	11101	
Set Q_0	00010	0011

Remainder = 0010=2 Quotient = 0011=3

16. Do the following arithmetic: (i) 25 – 19, (ii) –25 – 19, (iii) –25+19. [WBSCTE 2013]

Answer:

i) 25 = 011001, 19= 010011, –19= 101101, 25 –19= 25 + (–19)
= 011001 + 101101= 1000110 (discard carry 1)
000110 = 6

ii) 25 = 011001, –25 = 100111, 19 = 010011,–19=101101, (–25) + (–19) = 100111 + 101101 = 1010100 ,discarding carry and making 2's complement we get 101100 = 44 so answer –44

iii) 25= 011001, -25=100111, 19=010011, (–25)+19= 111010 (no carry make 2's complement and put '-' sign) ,so 2's complement of 111010 is 000110 and after putting '–' we get – 000110 = –6.

17. a) Describe Booth's algorithm with suitable block diagram and flowchart.

b) Show the steps of multiplication performed by using Booth's algorithm of 7×–5. [WBSCTE 2016, 2021]

Answer:

a) ***Refer to Article No. 2.6 of Unit at a glance.***

b)

Multiplicand –

Decimal: 7

Binary: 00000111

Multiplier –

Decimal: -5

Binary: 11111011

Two's Complement: 00000101

Steps –

Starting Out: 0000000000000111

Subtract: 0000010100000111

Shift: 0000001010000011

Shift: 0000000101000001

Shift: 0000000010100000

Add: 1111101110100000

Shift: 1111110111010000
Shift: 1111111011101000
Shift: 1111111101110100
Shift: 1111111110111010
Shift: 1111111111011101
Final Product (Binary):1111111111011101
Final Product (Decimal):-35

18. What is biased exponent? Represent 0.12 in floating point representation using 8 bit. What are direct & indirect addresses? **[WBSCTE 2017]**

Answer:

Bias exponent: In IEEE 754 floating point numbers, the **exponent** is **biased** in the engineering sense of the word – the value stored is offset from the actual value by the **exponent bias**. ... To calculate the **bias** for an arbitrarily sized floating point number apply the formula $2^{k-1} - 1$ where k is the number of bits in the **exponent**.
half-precision: 11 significant bits, 0.1 rounds to 0.0001100110011 in binary, which is 0.0999755859375 in decimal.

Direct addresses: Direct addressing means the instruction refers directly to the address being accessed. That is, the instruction encoding itself contains the address of thelocation. Depending on the instruction set, it may also allow computing a small index relative to the address.

Indirect addresses: Indirect addressing uses an address held in a register or other location to determine what memory location to read or write. The idea here is that the instruction itself isn't directly telling you the address to access, but rather indirectly telling the CPU where to find that address.

19. a) Convert and represent the decimal number 0.5624 in floating point representation using 8 bits.
b) Draw the Addition-Subtraction unit block diagram. **[WBSCTE 2018]**

Answer:

a) Decimal to binary:
0.5624 * 2 = 1.1248----------1
0.1248 * 2 = 0.2496----------0
0.2496 * 2 = 0.4992----------0
That is : $0.5624_{(10)} \rightarrow 0.100_{(2)}$
The scientific notation is: $0.001_{(2)} * 10^2$
$7+2 = 9_{(10)} = 0101_{(2)}$
Positive number so 0
Mantisa is 100
And the middle 4 digits are: exponent +7
So, the floating point number is:-
00101100

b) ***Refer to Article No. 2.4 of Unit at a glance.***

20. a) Divide Q = 11010 (dividend) by M = 110 (divisor) using restoring division method.
b) Show the steps of multiplication performed by using Booth's algorithm of 1011×10011. **[WBSCTE 2018]**

Answer:

a) A = 0
N = no. of bits in dividend = 5
M = divisor = 100
Q = dividend = 11010

A	Q	
00000	11010	initialize
00001	1010_	Shift AQ

A = A-M = 00001-100
= –011
= 10011
Q[0] = 1,

so, M = N–1 = 4

A	Q	
10011	1010_	initialize
00111	010_	Shift left AQ

A = A–M
= 00111–100
= 00011
Q[0] = 0
∴ Restore A
i.e., 00111
and M = N–1 = 3

A	Q	
00111	010_	initialize
01110	100_	Shift left AQ

A = A–M
= 01110–100
= 01010
Q[0] = 1
∴ M = N–1 = 2

A	Q	
01010	1000	initialize
10101	000_	Shift left AQ

A = A–M = 10101–100
=10001
Q[0] = 0
∴ Restore A
and M = N–1 = 1

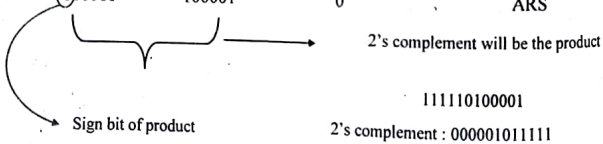
A	Q	
01010	0000	initialize
10100	000_	Shift left AQ

A = A–M = 10100–100
=10000
Q[0] = 0
∴ Quotient = 0
Reminder = 10000

b) *Booth's Algorithm*
1011×10011
Solution: Since, we need same no. of bits for both multiplicand and multiplier, so considering n = 6.
M= –5 = 111011
(–M) = 5 = 000101
Q = 19 = 010011

Trace Table

n	A	Q	q_0 (booth's bit)	Action
6	000000	010011	0	Initialization
	000101	010011	0	A=A-M
5	000010	101001	1	Arithmetic Right Shift
4	000001	010100	1	Arithmetic RS
	111100	010100	1	A=A+M
3	111110	001010	0	ARS
2	111111	000101	0	ARS
	000100	000101	0	A=A-M
1	000010	000010	1	ARS
	111101	000010	0	A=A+M
0	111110	100001	0	ARS



111110100001
2's complement : 000001011111
= 95

∴ Answer is (–95).

➲ **21. Represent $(12.625)_{10}$ in 32 bit floating point representation and what is odd parity checker?** [WBSCTE 2022]

Answer:
1st Part:
$12.625_{(10)}$ to 32 bit single precision IEEE 754 binary floating point (1 bit for sign, 8 bits for exponent, 23 bits for mantissa) = 0 10000010 10010100000000000000000
1. First, convert to the binary (base 2) the integer part: 12. Divide the number repeatedly by 2.
Keep track of each remainder.
We stop when we get a quotient that is equal to zero.
- division = quotient + remainder;
- 12 ÷ 2 = 6 + 0;
- 6 ÷ 2 = 3 + 0;
- 3 ÷ 2 = 1 + 1;
- 1 ÷ 2 = 0 + 1;

2. Construct the base 2 representation of the integer part of the number.
Take all the remainders starting from the bottom of the list constructed above.
$12_{(10)}$ =

$1100_{(2)}$

3. Convert to the binary (base 2) the fractional part: 0.625.
Multiply it repeatedly by 2.

Keep track of each integer part of the results.

Stop when we get a fractional part that is equal to zero.
- #) multiplying = integer + fractional part;
- 1) 0.625 × 2 = 1 + 0.25;
- 2) 0.25 × 2 = 0 + 0.5;
- 3) 0.5 × 2 = 1 + 0;

4. Construct the base 2 representation of the fractional part of the number.
Take all the integer parts of the multiplying operations, starting from the top of the constructed list above:
$0.625_{(10)}$ =

$0.101_{(2)}$

5. Positive number before normalization:
$12.625_{(10)}$ =

$1100.101_{(2)}$

6. Normalize the binary representation of the number.
Shift the decimal mark 3 positions to the left so that only one non zero digit remains to the left of it:

$12.625_{(10)}$ =

$1100.101_{(2)}$ =

$1100.101_{(2)} \times 2^0$ =

$1.1001\ 01_{(2)} \times 2^3$

7. Up to this moment, there are the following elements that would feed into the 32 bit single precision IEEE 754 binary floating point representation:
Sign: 0 (a positive number)

Exponent (unadjusted): 3

Mantissa (notnormalized): 1.1001 01

8. Adjust the exponent.
Use the 8 bit excess/bias notation:
Exponent (adjusted) =

Exponent (unadjusted) + $2^{(8-1)}$ - 1 =

$3 + 2^{(8-1)} - 1$ =

$(3 + 127)_{(10)}$ =

$130_{(10)}$

9. Convert the adjusted exponent from the decimal (base 10) to 8 bit binary.
Use the same technique of repeatedly dividing by 2:

- division = quotient + remainder;
- $130 \div 2 = 65 + 0$;
- $65 \div 2 = 32 + 1$;
- $32 \div 2 = 16 + 0$;
- $16 \div 2 = 8 + 0$;
- $8 \div 2 = 4 + 0$;
- $4 \div 2 = 2 + 0$;
- $2 \div 2 = 1 + 0$;
- $1 \div 2 = 0 + 1$;

10. Construct the base 2 representation of the adjusted exponent.
Take all the remainders starting from the bottom of the list constructed above:
Exponent (adjusted) =

$130_{(10)}$ =

$1000\ 0010_{(2)}$

11. Normalize the mantissa.
a) Remove the leading (the leftmost) bit, since it's allways 1, and the decimal point, if the case.
b) Adjust its length to 23 bits, by adding the necessary number of zeros to the right.
Mantissa (normalized) =

10 0101 0 0000 0000 0000 0000 =

100 1010 0000 0000 0000 0000

12. The three elements that make up the number's 32 bit single precision IEEE 754 binary floating point representation:
Sign (1 bit) =
0 (a positive number)

Exponent (8 bits) =
1000 0010

Mantissa (23 bits) =
100 1010 0000 0000 0000 0000

2nd Part:

Consider that a three bit message along with odd parity bit is transmitted at the transmitting end. Odd parity checker circuit receives these 4 bits and checks whether any error are present in the data. If the total number of 1s in the data is odd, then it indicates no error, whereas if the total number of 1s is even then it indicates the error since the data is transmitted with odd parity at transmitting end. The below figure shows the truth table for odd parity generator where PEC =1 if the 4-bit message received consists of even number of 1s (hence the error occurred) and PEC= 0 if the message contains odd number of 1s (that means no error).

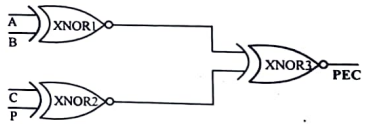
4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1

0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

The expression for the PEC in the above truth table can be simplified by K-map as

AB \ CP	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

After simplification, the final expression for the PEC is obtained as PEC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D). The expression for the odd parity checker can be designed by using three Ex-NOR gates as shown below.



22. a) Perform multiplication between 23 and 17 using fixed point multiplication algorithm.

b) What are the key characteristics of micro-programmed control? Explain different types of micro operation. **[WBSCTE 2022]**

Answer:

a) Multiplicand M = 23
Multiplier Q = 17

```
   10111
  x10001
  -------
   10111
  00000
 00000
00000
10111
---------
110000111
```

Product P = 391

b) 1st Part:

Characteristics of micro programmed control units:

- These control units are implemented as micro programs of routines.
- The control unit implemented in micro program is implemented in the form of a CPU inside another CPU.
- These types of circuits are simple but comparatively slower.

2nd Part:

The micro-operations in digital computers are of 4 types:

1. Register transfer micro-operattions transfer binary information from one register to another.
2. Arithmetic micro-operations perform arithmetic operations on numeric data stored in registers.
3. Logic micro-operations perform bit manipulation ioperation on non-numeric data stored in registers.
4. Shift mocro-operations perform shift micro-ioperations performed on data.

Arithmetic Micro-operations

In general, the Arithmetic Micro-operations deals with the operations performed on numeric data stored in the registers.

The basic Arithmetic Micro-operations are classified on the following categories:

1. Addition
2. Substruction
3. Increment
4. Decrement
5. Shift

Some additional Arithmetic Micro-operations are classified as:

1. Add with carry
2. Substract with borrow
3. Transfet/Load, etc.

The following table shows the symbolic representation of various Arithmetic Micro-operations.

Symbolic Representation	Description
R3 ← R1 + R2	The contents of R1 plus R2 are transferred to R3.
R3 ← R1 - R2	The contents of R1 minus R2 are transferred to R3.
R2 ← R2'	Complete the contents of R2 (1's complement)
R2 ← R2' + 1	2's complement the content of R2 (negate)
R3 ← R1 + R2' + 1	R1 plus the 2's complement of R2 (substruction)
R1 ← R1 + 1	Incteremt the contents of R1 by one
R1 ← R1 – 1	Decrement the contents of R1 by one

Logic Micro-Operations

These are binary micro-operations performed on the bits stored in the registers.
These operations consider each bit separately and treat them as binary variables.
Let us consider the X-OR micro-operation with the contents of two registers R1 and R2.

P: R1 ← R1 X-OR R2

In the above statement we have also include a Control Function.
Assume that each r4egister has 3 bits. Let the content of R1 be **010** and R2 be **100.**
The X-OR micro-operation will be:

010 → R1
100 → R2
110 → R1 after P = 1

Shift Micro-Operations

These are used for serial transfer of data. Tha means we can shift the contents of the register to the left or right. In the **shift left** operation the serial input transfers a bit to the right most position and in **shift right** operation the serial input transfers a bit to the left most position.
There are three types of shifts as follows:

a) Logical shift

it transfers 0 through the serial input. The symbol **"shl"** is used for logical shift left and **"shr"** is used for logical shift right.

R1 ← she R1
R1 ← she R1

The register symbol must be same on both sides of arrows.

b) Circular shift

This circulates or rotates the bits of register around the two ends without any loss of data or contents. In this, the serial output of the shift register is connected to its serial input. "cil" and "cir" is used for circular shift keft and right respectively.

c) Arithmetic shift

this shift a signed binary number to left or right. An **arithmetic shift left** multiplies a signed binary number by 2 and **shift left** divides the number by 2. Arithmetic shift micro-

operation leaves the sign bit unchanged becaused the signed number remains same when it is multiplied or divided by 2.

➲ 23. What are the advantages of microprogramming control over hardwired control? **[Model Question]**

Answer:

Advantages: As, once written, the microprograms can be changed, hence in case of microprogrammed control unit, the control memory design and implementation can be changed as needed and is very flexible.
But in case of hardwired unit to change, the entire hardware configurations are to be changed and hence such control units are not very flexible and changeable.
Disadvantages: Hardwired control units (as hardware implemented) are faster than microprogrammed control unit (software implemented).

➲ 24. What do you mean by instruction cycle, machine cycles and T states? **[Model Question]**

Answer:

Instruction cycle is defined, as the time required completing the execution of an instruction.
Machine cycle is defined as the time required completing one operation of accessing memory, I/O or acknowledging an external request.
Tcycle is defined as one subdivision of the operation performed in one clock period.

➲ 25. What is meant by stored program organization? **[Model Question]**

Answer:

In a stored program organization, there is a memory that stores the programs (i.e. set of instructions) and data (i.e. operands), a simple processor register (say, an accumulator) and an instruction format with an opcode part and an operand (or address) part. The operand (data) is fetched from the memory and operated together (as per the opcode) with the data stored in the processor register.

➲ 26. Explain the basic ideas of program execution. **[Model Question]**

Answer:

The basic ideas of program execution are as follows:
(a) A Computer executes users' programs.
(b) A user writes his program as a meaningful sequence of instructions. These instructions must belong to the Instruction set of the computer.
(c) The computer is capable of executing each and every instruction in its instruction set.
(d) A user's program is stored in the main memory.
(e) The starting address of the program is first loaded into the PC.
(f) The first instruction is first fetched from the main memory to the CPU. Then the instruction is executed in the CPU. This execution may require additional memory

accesses (i.e. memory read or memory write operations) in order to fetch the operands from the memory if needed.

(g) After executing the first instruction, the CPU next fetches and executes the second instruction. This way, the CPU fetches and executes the remaining instructions in the program and thus completes the execution of the entire program.

27. What are the operations that can be performed to execute an instruction? [Model Question]

Answer:

Generally an instruction can be executed by performing one or more of the following operations in some specified sequence:

(a) A word or data is transferred from one processor register to another or to the ALU.

(b) An arithmetic or logic operation is performed and the result is stored in a processor register.

(c) Contents of a given memory location is fetched and stored into a processor register.

(d) A word is stored from a processor register into a given memory location.

28. RISC architecture is more suitable for pipeline implementation-explain. [WBSCTE 2014, 2019]

OR,

Write down the features of RISC architecture. [WBSCTE 2019]

Answer:

Among the characteristics attributed to RISC is its ability to use an efficient instruction pipeline. The simplicity of the instruction set can be utilized to implement an instruction pipeline using a small number of suboperations, with each being executed in one clock cycle. Because of the fixed-length instruction format, the decoding of the operation can occur at the same time as the register selection. All data manipulation instructions have register-to-register operations. Since all operands are in registers, there is no need for calculating an effective address or fetching of operands from memory. Therefore, the instruction pipeline can be implemented with two or three segments. One segment fetches the instruction from program memory, and the other segment executes the instruction in the ALU. A third segment may be used to store the result of the ALU operation in a destination register. The data transfer instructions in RISC are limited to load and store instructions. These instructions use register indirect addressing. They usually need three or four stages in the pipeline. To prevent conflicts between a memory access to fetch an instruction and to load or store an operand, most RISC machines use two separate buses with two memories: one for storing the instructions and the other for storing the data. The two memories can sometime operate at the same speed as the CPU clock and are referred to as cache memories.

29. Discuss Flynn's classification in details. [WBSCTE 2014, 2015, 2016, 2019]

Answer:

First of all as we know the CPU comprises of The Registers, The ALU Unit and the Control Unit. Processing of the data is done in CPU Registers (an area or a memory where processing takes place).

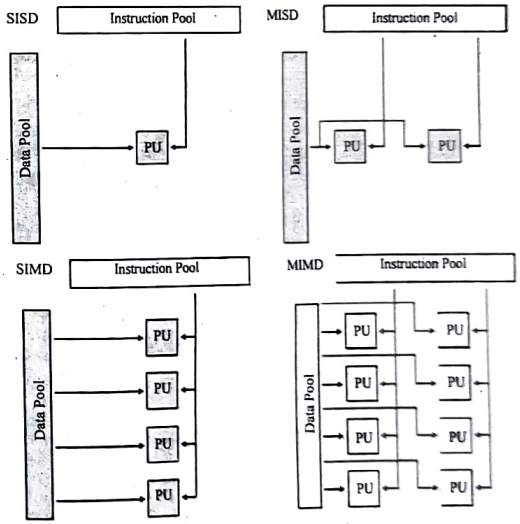
M. J. Flynn suggested these architectures for enhancing the computational speed of the computer:

1. Single Instruction Single Data (SISD): Single instruction is performed on a single set of data in a sequential form. Most of our computers today are based on this architecture. Von Neumann fits into this category. Data is executed in a sequential fashion (one by one).

2. Single Instruction Multiple Data (SIMD): Single Instruction is performed on multiple data. A good example is the 'For' loop statement. Over here instruction is the same but the data stream is different.

3. Multiple Instruction Single Data (MISD): Over here N no. of processors are working on different set of instruction on the same set of data. There is no commercial computer of this kind also these are used in Space Shuttle controlling computer (all the buttons you must have noticed in the control center).

4. Multiple Instruction Multiple Data (MIMD): Over here there is an interaction of N no. of processors on a same data stream shared by all processors. Now over here if you have noticed a lot of computers connected to each other and when they perform a task on the same data (data is then shared).If the processing is high it is called Tightly Coupled and Loosely Coupled vice-versa. Most Multiprocessor fit into this category.



30. Explain instruction pipeline with suitable example and diagram. [WBSCTE 2014, 2019]

Answer:

Instruction Pipelining:

To extract better performance instruction execution can be done through instruction pipeline. The instruction pipelining involves decomposing of an instruction execution to a number of pipeline stages.

Some of the common pipeline stages can be instruction fetch (IF), instruction decode (ID), operand fetch (OF), execute (EX), store results (SR).

An instruction pipe may involve any combination of such stages. A major design decision here is that the instruction stages should be of equal execution time. The reason why it should be is,

A pipeline allows overlapped execution of instructions. Thus, during the course of execution of an instruction the following may be a scenario of execution.

A pipeline allows overlapped execution of instructions. Thus, during the course of execution of an instruction the following may be a scenario of execution.

Time Slot –>	1	2	3	4	5	6	7	8	9	10	11
Instruction 1	IF	ID	OF	EX	SR						
Instruction 2		IF	ID	OF	EX	SR					
Instruction 3			IF	ID	OF	EX	SR				
Instruction 4				IF	ID	OF	EX	SR			
Instruction 5					IF	ID	OF	EX	SR		
Instruction 6						IF	ID	OF	EX	SR	
Instruction 7							IF	ID	OF	EX	SR

Instruction Pipeline

Please note the following observations about the above figure:

- The pipeline stages are like steps. Thus, a step of the pipeline is to be complete in a time slot. The size of the time slot will be governed by the stage taking maximum time. Thus, if the time taken in various stages is almost similar, we get the best results.
- The first instruction execution is completed on completion of 5th time slot, but afterwards, in each time slot the next instruction gets executed. So, in ideal conditions one instruction is executed in the pipeline in each time slot.
- After the 5th time slot and afterwards the pipe is full. In the 5th time slot the stages of execution of five instructions are:

SR (instruction 1)	(Requires memory reference)
EX (instruction 2)	(No memory reference)
OF (instruction 3)	(Requires memory reference)
ID (instruction 4)	(No memory reference)
IF (instruction 5)	(Requires memory reference)

31. Explain different pipeline hazards and their possible solution. [WBSCTE 2015]

Answer: *Refer to Article No. 2.15 of Unit at a glance.*

32. What do you mean by pipeline hazards/conflicts? Discuss the different types of hazards being observed and also explain the possible solutions. [WBSCTE 2016, 2021]

Answer:

1st Part:

A pipeline hazard refers to a situation in which a correct program ceases to work correctly due to implementing the processor with a pipeline. There are three fundamental types of hazard: data hazards, branch hazards, and structural hazards.

2nd Part: *Refer to Article No. 2.15 of Unit at a glance.*

33. What is instruction pipeline? Explain it with the flowchart. [WBSCTE 2017]

Answer:

An instruction pipeline is a technique used in the design of computers system to increase their instruction throughput. The fundamental idea is to split the processing of a computer instruction into a series of independent steps, with storage at the end of each step. The principles used in instruction pipelining can be used in order to improve the performance of computers in performing arithmetic operations such as add, subtract, and multiply.

In a program there is several numbers of instructions. There are five steps to execute an instruction and the steps are:

i) Fetch
ii) Decode
iii) Operand fetch
iv) Execute
v) Write back

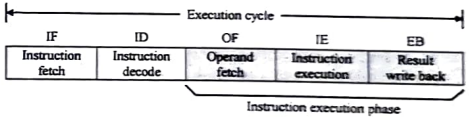
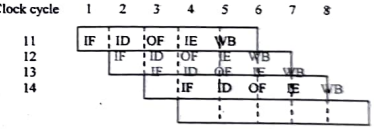


Fig: Instruction pipeline stages

The streams of instructions are executed in the pipeline in an overlapped manner.



Pipelining can be applied to arithmetic operations. As an example, we show a floating-point add pipeline in Figure below. The floating-point add unit has several stages:

Unpack	Align	Add	Normalize

Fig: Floating point add pipeline stages

- ***Unpack:*** The unpack stage partitions the floating-point numbers into the three field: the sign field, exponent field, and mantissa field. Any special cases such as not-a-number (NaN), zero, and infinities are detected during this stage.
- ***Align:*** This stage aligns the binary points of the two mantissas by right-shifting the mantissa with the smaller exponent.
- ***Add:*** This stage adds the two aligned mantissas.
- ***Normalize:*** This stage packs the three fields of the result after normalization and rounding into the IEEE-754 floating-point format. Any output exceptions are detected during this stage.

➲ 34. Write short note on Space time diagram? **[WBSCTE 2019]**

Answer:

Space time diagram:

A task is the total operation performed going through all segment of pipeline. The behavior of a pipeline can be illustrated with a space time diagram. This shows the segment utilization as a function of time.

Space time diagram for 4 segments and 6 tasks:

Clock cycle → Segment ↓	1	2	3	4	5	6	7	8	9
1	T1	T2	T3	T4	T5	T6			
2		T1	T2	T3	T4	T5	T6		
3			T1	T2	T3	T4	T5	T6	
4				T1	T2	T3	T4	T5	T6

➲ 35. a) What are the major characteristics of RISC architecture?
b) What do you mean by speedup ratio? What are the reasons for which theoretical maximum speedup cannot be obtained in reality? **[WBSCTE 2021]**

Answer:

a) RISC meaning reduced instruction set as the acronym say aims to reduce the execution times of instructions by simplifying the instructions. The major characteristics of RISC are as follows:

- Compared to normal instructions they have a lower number of instructions.
- The addressing modes in case of RISC is also lower.
- All the operations that are required to be performed take place within the CPU.
- All instruction are executed in a single cycle hence have a faster execution time.
- in this architecture the processors have a large number of registers and a much more efficient instruction pipeline.
- Also the instruction formats are of fixed length and can be easily decoded.

b) 1st Part:

Speed up ratio (*S*): It is defined as the speedup of a pipeline processing with respect to the equivalent non-pipeline processing.

Formula for calculating Speed up ratio is as shown:

$$S = \frac{nt_n}{(k+n-1)t_p}$$

Number of tasks $n = 100$

For Non-pipeline:

Time taken by non-pipeline to process a task $t_n = 50ns$

Total time taken by non-pipeline to process 100 task=nt_n

=100×50

=5000ns

For pipeline:

Number of segment pipeline $k = 6$

Time period of 1 clock cycle $t_p = 10ns$

Total time requires to complete n tasks in k segment pipeline with t_p clock cycle time:

$=(k+n-1)t_p$

= (6+100-1)10

=1050ns

Speed up ratio:

When total time taken by the pipeline to process 100 tasks is divided by the total time requires to complete n tasks is k segment pipeline with tp clock cycle time then speed up ratio is obtained.

$$S = \frac{5000}{1050} = 4.76$$

2nd Part:

The following are various limitations due to which any pipeline system cannot operate at its maximum theoretical rate i.e., k (speed up ratio).

a. Different segments take different time to complete there suboperations, and in pipelining clock cycle must be chosen equal to time delay of the segment with maximum propagation time. Thus all other segments have to waste time waiting for next clock cycle. The possible solution for improvement here can if possible subdivide the segment into different stages i.e., increase the number of stages and if segment is not subdivisible than use multiple of resource for segment causing maximum delay so that more than one instruction can be executed in to different resources and overall performance will improve.

b. Additional time delay may be introduced because of extra circuitry or additional software requirement is needed to overcome various hazards, and store the result in the intermediate registers. Such delays are not found in non pipeline circuit.

c. Further pipelining can be of maximum benefit if whole process can be divided into suboperations which are independent to each other. But if there is some resource conflict or data dependency i.e., a instruction depends on the result of pervious instruction which is not yet available than instruction has to wait till

result become available or conditional or non conditional branching i.e., the bubbles or time delay is introduced.

36. Consider the five stage pipelined processor specified by the following reservation table:

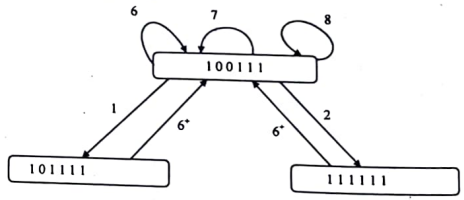
	1	2	3	4	5	6
S_1	x					x
S_2		x			x	
S_3			x			
S_4				x		
S_5		x				x

(a) What are the forbidden latencies and the initial collision vector?
(b) Draw the state transition diagram for scheduling the pipeline.
(c) Determine all simple cycles, greedy cycle and MAL.
(d) What will be the maximum throughput of the pipeline? **[Model Question]**

Answer:

(a) The forbidden latencies, F= 0,3, 4, 5
The initial collision vector, V= [1,0,0,1,1,1]

(b) State 1: 100111
Reaches state 2 (101111) after 1 cycle
Reaches state 3 (111111) after 2 cycle
Reaches state 1 (100111) after 6 cycle or more cycles
State 2: 101111
Reaches state 3 (111111) after 1 cycle
Reaches state 1 (100111) after 6 cycle or more cycles
State 3: 111111
Reaches state 1 (100111) after 6 cycle or more cycles



(c) The simple cycles are : (1,6) , (2,6), (6), (7), (8), (1,6,6), (1,6,7), (1,6,8), (2,6,6), (2,6,7), (2,6,8)
The Greedy cycle is: (1,1,6)
MAL= 2.66

(d) The maximum throughput = 0.375 instruction per cycle

37. Distinguish a static pipeline from a dynamic pipeline. **[Model Question]**

Answer:

In general, a static pipeline is one, which follows a fixed series of steps. Traditional linear pipeline are static pipelines because they are used to perform fixed functions. The given function is partitioned into a sequence of linearly ordered static sub functions. In CPU design, static pipelines are easier to design and less expensive to manufacture.
The alternative is a dynamic pipeline, where the steps are determined by a set of rules. A dynamic pipeline can be reconfigured to perform variable functions at different times. It allows feed forward and feedback connections in addition to streamline connections. In CPU design, dynamic pipelines can significantly boost performance with out of order execution.

38. Describe the time stationary and data stationary control schemes. **[Model Question]**

Answer:

A control scheme determines how a segment decides what operation to perform on each transaction and where to send the transaction when it is done in the segment. The specification for the control scheme says that if every transaction goes to the correct segments and each segment performs the correct operation, then every transaction will eventually request to exit the pipeline. A segment can decide what operation it is to perform on a transaction based on *data-stationary* and/or *time-stationary* control information. Data-stationary control information is carried along with the transaction as it proceeds between segments. Time-stationary information is provided to the segment at each time cycle, regardless of what transaction is in the segment. A segment can determine where to send a transaction when it is done based on the data-stationary information with which the transaction entered the segment, data-dependent information generated within the segment (such as the result of a comparison or an overflow condition), and/or the current value of time-stationary control signals.

39. (a) A 30% enhancement in speedup for a component of the processor has been proposed for a new architecture. If the enhancement is usable only for 50% for the time, what is fraction of the time must enhancement be used to achieve an overall speedup of 10?
(b) What are the different approaches taken by pipeline processor to handle branch instructions? Briefly illustrate any two approaches? **[Model Question]**

Answer:

(a) Say, the fraction of the time must enhancement is used to achieve an overall speedup $= x$
So, x. $(30/50)=10 \Rightarrow x= 16.6$

(b) One of the major problems in designing an instruction pipeline is assuring a steady flow of instructions to initial stages of the pipeline. However, 15-20% of instructions in an assembly-level stream are (conditional) branches. Of these, 60-70% take the branch to

a target address. Until the instruction is actually executed, it is impossible to determine whether the branch will be taken or not.
A number of techniques can be used to minimize the impact of the branch instruction (the branch penalty).

- ***Multiple streams***
  - 0 Replicate the initial portions of the pipeline and fetch both possible next instructions
  - 1 Increases chance of memory contention
  - 2 Must support multiple streams for each instruction in the pipeline
- ***Prefetch branch target***
  - 3 When the branch instruction is decoded, begin to fetch the branch target instruction and place in a second prefetch buffer
  - 4 If the branch is not taken, the sequential instructions are already in the pipe, so there is not loss of performance
  - 5 If the branch is taken, the next instruction has been prefetched and results in minimal branch penalty (don't have to incur a memory read operation at the end of the branch to fetch the instruction)
- ***Loop buffer: Look ahead, look behind buffer***
  - 6 Many conditional branches operations are used for loop control
  - 7 Expand prefetch buffer so as to buffer the last few instructions executed in addition to the ones that are waiting to be executed
  - 8 If buffer is big enough, entire loop can be held in it, this can reduce the branch penalty.
- ***Branch prediction***
  - 9 Make a good guess as to which instruction will be executed next and start that one down the pipeline.
  - 10 Static guesses: make the guess without considering the runtime history of the program : Branch never taken, Branch always taken, Predict based on the opcode
  - 11 Dynamic guesses: track the history of conditional branches in the program.
- ***Delayed branch***
  - 12 Minimize the branch penalty by finding valid instructions to execute in the pipeline while the branch address is being resolved.
  - 13 It is possible to improve performance by automatically rearranging instruction within a program, so that branch instruction occur later than actually desired
  - 14 Compiler is tasked with reordering the instruction sequence to find enough independent instructions to feed into the pipeline after the branch that the branch penalty is reduced to zero.

➲ **40. Compare superscalar, superpipeline and superscalar superpipelined architecture.** **[Model Question]**

Answer:

Superscalar

Superscalar processing has multiple functional units are kept busy by multiple instructions. As execution pipelines have approached the limits of speed, parallel execution has been required to improve performance. Super-pipelined machines can issue only one instruction per cycle, but they have cycle times shorter than the latency of any functional unit. In some cases superscalar machines still employ a single fetch-decode-dispatch pipe that drives all of the units. For example, the Ultra SPARC splits execution after the third stage of a unified pipeline. However, it is becoming more common to have multiple fetch-decode-dispatch pipes feeding the functional units. Superscalar operation is limited by the number of independent operations that can be extracted from an instruction stream.

Super-pipeline

Given a pipeline stage time T, it may be possible to execute at a higher rate by starting operations at intervals of T/n. This can be accomplished in two ways:

1. Further divide each of the pipeline stages into n substages.
2. Provide n pipelines that are overlapped.

The first approach requires faster logic and the ability to subdivide the stages into segments with uniform latency. The second approach could be viewed in a sense as staggered superscalar operation, and has associated with it all of the same requirements except that instructions and data can be fetched with a slight offset in time.
Super-pipelining is limited by the speed of logic, and the frequency of unpredictable branches. Stage time cannot productively grow shorter than the inter stage latch time, and so this is a limit for the number of stages. The MIPS R4000 is sometimes called a super-pipelined machine. The benefit of such extensive pipelining is really only gained for very regular applications such as graphics.

Superscalar-Super-pipeline

We may also combine superscalar operation with super-pipelining and the result is potentially the product of the speedup factors. However, it is even more difficult to interlock between parallel pipes that are divided into many stages. Also, the memory subsystem must be able to sustain a level of instruction throughput corresponding to the total throughput of the multiple pipelines – stretching the processor/memory performance gap even more. Of course, with so many pipes and so many stages, branch penalties become huge, and branch prediction becomes a serious bottleneck.
But the real problem may be in finding the parallelism required to keep all of the pipes and stages busy between branches. Consider that a machine with 12 pipelines of 20 stages must always have access to a window of 240 instructions that are scheduled so as to avoid all hazards, and that the average of 40 branches that would be present in a block of that size are all correctly predicted sufficiently in advance to avoid stalling in the prefetch unit.

➲ 41. Give an example of concurrent processes in computation? [Model Question]

Answer:

There is a prime number generator pipeline as shown in figure below. Integer numbers are generated and successively tested for divisibility by previously generated primes in a linear pipeline of primes. The circle numbers represent those being generated. A number enters the pipeline from the left end and is eliminated if it is divisible by the prime number tested at a pipeline stage. All the numbers being forwarded to the right of a pipeline stage are those indivisible by all the prime numbers tested on the left of that stage.

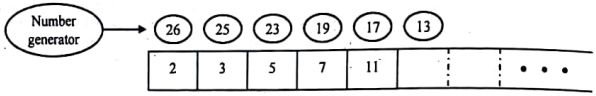


Fig: (a) Pipeline concurrency

➲ 42. Write short notes of the following:

a) Array Processor [WBSCTE 2019]
b) Vector Processing [WBSCTE 2021]
b) Reservation Table [Model Question]
c) Branch handling in instruction pipeline [Model Question]
d) Amdahl's law and its significance [Model Question]

Answer:

a) *Refer to 2.18 and 2.19 of Unit at a glance.*

b) Vector Processing

Vector processing is a central processing unit that can perform the complete vector input in individual instruction. It is a complete unit of hardware resources that implements a sequential set of similar data elements in the memory using individual instruction.

The scientific and research computations involve many computations which require extensive and high-power computers. These computations when run in a conventional computer may take days or weeks to complete. The science and engineering problems can be specified in methods of vectors and matrices using vector processing.

Features of Vector Processing

There are various features of Vector Processing which are as follows –

- A vector is a structured set of elements. The elements in a vector are scalar quantities. A vector operand includes an ordered set of n elements, where n is known as the length of the vector.
- Each clock period processes two successive pairs of elements. During one single clock period, the dual vector pipes and the dual sets of vector functional units allow the processing of two pairs of elements.

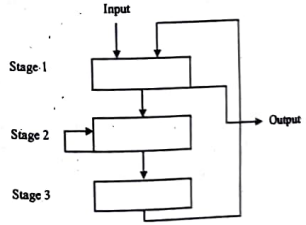
  As the completion of each pair of operations takes place, the results are delivered to appropriate elements of the result register. The operation continues just before

the various elements processed are similar to the count particularized by the vector length register.

- In parallel vector processing, more than two results are generated per clock cycle. The parallel vector operations are automatically started under the following two circumstances –
  - When successive vector instructions facilitate different functional units and multiple vector registers.
  - When successive vector instructions use the resulting flow from one vector register as the operand of another operation utilizing a different functional unit. This phase is known as chaining.
- A vector processor implements better with higher vectors because of the foundation delay in a pipeline.
- Vector processing decrease the overhead related to maintenance of the loop-control variables which creates it more efficient than scalar processing.

b) Reservation Table:

There are two types of pipelines: static and dynamic. A static pipeline can perform only one function at a time, whereas a dynamic pipeline can perform more than one function at a time. A pipeline reservation table shows when stages of a pipeline are in use for a particular function. Each stage of the pipeline is represented by a row in the reservation table. Each row of the reservation table is in turn broken into columns, one per clock cycle. The number of columns indicates the total number of time units required for the pipeline to perform a particular function. To indicate that some stage S is in use at some time ty, an X is placed at the intersection of the row and column in the table corresponding to that stage and time. Figure below represents a reservation table for a static pipeline with three stages. The times t0, t1, t2, t3, and t4 denote five consecutive clock cycles. The position of X's indicate that, in order to produce a result for an input data, the data must go through the stages 1, 2, 2, 3, and 1, progressively. As shown later in this section, the reservation table can be used for determining the time difference between input data initiations so that collisions won't occur. (Initiation of an input data refers to the time that the data enter the first stage of the pipeline).



	t_0	t_1	t_2	t_3	t_4
Stage1	X				X
Stage2			X	X	
Stage3				X	

c) Branch handling in instruction pipeline:

In pipeline process the branch instructions are those that tell the processor to make a decision about what the next instruction to be executed should be based on the results of

another instruction. Branch instructions can be troublesome in a pipeline if a branch is conditional on the results of an instruction which has not yet finished its path through the pipeline.

For example:

```
Loop : add   $r3,   $r2,   $r1
       sub   $r6,   $r5,   $r4
       equal $r3, $r6,  Loop
```

The example above instructs the processor to add r1 and r2 and put the result in r3, then subtract r4 from r5, storing the difference in r6. In the third instruction, beq stands for branch if equal. If the contents of r3 and r6 are equal, the processor should execute the instruction labeled "Loop." Otherwise, it should continue to the next instruction. In this example, the processor cannot make a decision about which branch to take because neither the value of r3 or r6 have been written into the registers yet.

The processor could stall, but a more sophisticated method of dealing with branch instructions is branch prediction. The processor makes a guess about which path to take - if the guess is wrong, anything written into the registers must be cleared, and the pipeline must be started again with the correct instruction. Some methods of branch prediction depend on stereotypical behavior. Branches pointing backward are taken about 90% of the time since backward-pointing branches are often found at the bottom of loops. On the other hand, branches pointing forward are only taken approximately 50% of the time. Thus, it would be logical for processors to always follow the branch when it points backward, but not when it points forward. Other methods of branch prediction are less static: processors that use dynamic prediction keep a history for each branch and use it to predict future branches. These processors are correct in their predictions 90% of the time.

d) Amdahl's law and its significance:

Amdahl's law is a model for the relationship between the expected speedup of parallelized implementations of an algorithm relative to the serial algorithm, under the assumption that the problem size remains the same when parallelized. For example, if for a given problem size a parallelized implementation of an algorithm can run 12% of the algorithm's operations arbitrarily quickly (while the remaining 88% of the operations are not parallelizable), Amdahl's law states that the maximum speedup of the parallelized version is $1/(1-0.12) = 1.136$ times as fast as the non-parallelized implementation.

More technically, the law is concerned with the speedup achievable from an improvement to a computation that affects a proportion P of that computation where the improvement has a speedup of S. (For example, if 30% of the computation may be the subject of a speed up, P will be 0.3; if the improvement makes the portion affected twice as fast, S will be 2.) Amdahl's law states that the overall speedup of applying the improvement will be:

$$\frac{1}{(1-P)+\frac{P}{S}}$$

To see how this formula was derived, assume that the running time of the old computation was 1, for some unit of time. The running time of the new computation will be the length of time the unimproved fraction takes (which is $1 - P$), plus the length of time the improved fraction takes. The length of time for the improved part of the computation is the length of the improved part's former running time divided by the speedup, making the length of time of the improved part P/S. The final speedup is computed by dividing the old running time by the new running time, which is what the above formula does.

Unit 3: Introduction to Microprocessor Architecture

Unit at a glance:

3.1 Instruction Format

Instruction, in a computer, specifies an operation to be carried out and the set of operands or data to be used for that purpose. Instruction format deals with the looks of a basic instruction. Each instruction has three parts.

Opcode: Opcode is the operation code that indicates the specific operation to be carried out by the instruction. The operation field of an instruction is called the opcode.

Operands: Operand fields contain the addresses of storage locations of the arguments to be used in the operation.

Operand Address: Each operation specified by computer instructions are executed on some data stored in memory or processor registers. These data or operands are specified either by memory address or by register address.

3.1.1 Types of Instruction Code Formats

(i) Memory-reference instruction: Instructions that need reference to a memory location (i.e., reference to a memory register is needed).

Example:

LOAD 1001 ⇒ The content of memory location 1001 is loaded to the accumulator

(ii) Register-reference instruction:

Instructions, which reference a CPU register for the purpose of addressing. An operand from memory (i.e. no memory reference) is not needed.

Example:

ADD_{Direct} B ⇒ The content of CPU register B is added to the accumulator.

(iii) Input-output instruction: Instructions, which reference an I/O register for the purpose of addressing. Here also no memory references are needed.

Example:

IN A, 2010 ⇒ The I/O unit may have numerous registers in it. The content of register 2010 is to be loaded inside the accumulator.

3.2 Intel 8086 microprocessor

8086 Microprocessor is an enhance version of 8085 Microprocessor that was designed by intel in 1976. It is a 16-bit Microprocessor having 20 address lines and 16 data lines that provides up to 1 MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

3.2.1 Features of 8086

The most prominent features of a 8086 microprocessor are as follows –

- It has an instruction queue, which is capable of storing six instruction bytes from the memory resulting in faster processing.
- It was the first 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus resulting in faster processing.
- It is available in 3 versions based on the frequency of operation –
  - 8086 → 5 MHz
  - 8086-2 → 8 MHz
  - (c)8086-1 → 10 MHz
- It uses two stages of pipelining, i.e. Fetch stage and Execute Stage, which improves performance.
- Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.
- Execute stage executes these instructions.
- It has 256 vectored interrupts.
- It consists of 29,000 transistors.

3.2.2 Comparison between 8085 & 8086 Microprocessor

- **Size** – 8085 is 8-bit microprocessor, whereas 8086 is 16-bit microprocessor.
- **Address Bus** – 8085 has 16-bit address bus while 8086 has 20-bit address bus.
- **Memory** – 8085 can access up to 64 KB, whereas 8086 can access up to 1 MB of memory.
- **Instruction** – 8085 doesn't have an instruction queue, whereas 8086 has an instruction queue.
- **Pipelining** – 8085 doesn't support a pipelined architecture while 8086 supports a pipelined architecture.
- **I/O** – 8085 can address 2^8 = 256 I/O's, whereas 8086 can access 2^16 = 65,536 I/O's.
- **Cost** – The cost of 8085 is low whereas that of 8086 is high.

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. Instruction in computer languages consists of– [WBSCTE 2011, 2015]
(a) OPCODE (b) OPERAND
(c) Both of above (d) None of the above
Answer: (c)

2. In Which addressing mode is operand specified in the instruction itself? [WBSCTE 2018, 2021]
(a) Register mode (b) Immediate mode
(c) Direct Address mode (d) Index Addressing mode
Answer: (b)

3. The number of fetch operation to execute instruction in immediate mode is [WBSCTE 2021]
(a) 0 (b) 1 (c) 2 (d) none of these
Answer: (a)

4. The instruction LOAD is a [WBSCTE 2021]
(a) zero-address instruction (b) one-address instruction
(c) two-address instruction (d) three-address instruction
Answer: (b)

5. The instruction 1111 111100001100 is a [WBSCTE 2021]
(a) direct memory reference instruction
(b) indirect memory reference instruction
(c) register reference instruction
(d) input output instruction
Answer: (a)

6. An interrupt that can be temporarily ignored is [WBSCTE 2022]
(a) Vectored interrupt (b) Non-maskable interrupt
(c) Maskable interrupt (d) High priority interrupt
Answer: (c)

7. The 16 bit flag of 8086 microprocessor is responsible to indicate _____ [Model Question]
(a) The condition of result of ALU operation (b) The condition of memory
(c) The result of addition (d) The result of subtraction
Answer: (a)

8. The BIU contains FIFO register of size _____ bytes. [Model Question]
(a) 8 (b) 6 (c) 4 (d) 12
Answer: (b)

9. The BIU prefetches the instruction from memory and store them in _____ [Model Question]
(a) Queue (b) Register (c) Memory (d) Stack
Answer: (a)

10. The 8086 fetches instruction one after another from _____ of memory. [Model Question]
(a) Code segment (b) IP (c) ES (d) SS
Answer: (a)

B. Fill in the blanks in the following statements:

11. Instruction register stores ***the instruction currently being executed***. [WBSCTE 2022]

12. I/O address in 8086 is ***16*** bit. [WBSCTE 2022]

13. The intel 8086 microprocessor is a ***16*** processor. [Model Question]

14. In 8086 microprocessor, the address bus is ***30*** bit wide. [Model Question]

C. Answer the following questions:

15. What is Opcode? Give example. [WBSCTE 2013, 2018]
Answer:
An opcode is the first byte of an instruction in machine language which tells the hardware what operation needs to be performed with this instruction. Every processor/controller has its own set of opcodes defined in its architecture. An opcode is followed by data like address, values etc if needed.

16. What do you mean by instruction format? [WBSCTE 2014, 2016]
Answer:
The instruction format is simply a sequence of bits (binary 0 or, 1) contained in a machine instruction that defines the layout of the instruction.

17. What do you mean by Effective Address? [WBSCTE 2021]
Answer:
An effective address is the location of an operand which is stored in memory.

18. Give an example of a 4 bit, 8 bit, 16 bit and 32 bit microprocessor. [WBSCTE 2022]

Answer:
4-bit Processor - TMS 1000 ;8-bit Processor - **8085 / Z80 / 6800**; 16-bit Processor - 8086 / 68000 / Z8000; 32-bit Processor - 80386 / 80486.

19. What is interrupt? [WBSCTE 2022]

Answer:
Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.

20. What's the difference between interrupt service routine and subroutine? [WBSCTE 2022]

Answer:
A subroutine is called by a program instruction to perform a function needed by the calling program. An interrupt-service routine is initiated by an event such as an input operation or a hardware error.

21. Mention the different varieties of 8086 and their corresponding speeds. [Model Question]

Answer:
The following shows the different varieties of 8086 available and their corresponding speeds.

Types	Speeds
8086	5 MHz
8086-1	10 MHz
8086-2	8 MHz

22. Mention the kind operations possible with 8086. [Model Question]

Answer:
It can perform bit, byte, word and block operations. Also multiplication and division operations can be performed by 8086.

23. Mention the total number of registers of 8086 and show the manner in which they are grouped. [Model Question]

Answer:
There are in all fourteen numbers of 16-bit registers. The different groups are made as hereunder:

- Data group, pointers and index group, status and control flag group and segment group.
- The data group consists of AX (accumulator), BX (base), CX (count), and DX (data).
- Pointer and Index group consists of SP (Stack pointer), BP (Base Pointer), SI (Source Index), DI (Data Segment) and IP (Instruction Pointer).
- Segment group consists of ES (Extra Segment), CS (Code Segment), DS (Data Segment) and SS (Stack Segment).

- Control flag consists of a single 16-bit flag register.

Figure below shows the registers placed in the different groups to form a programming model.

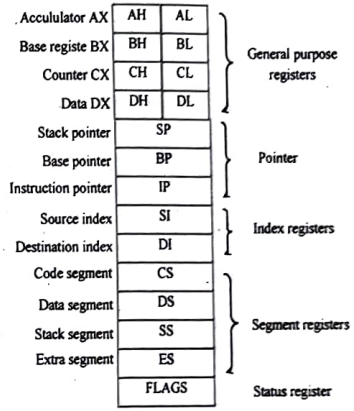


Fig: Schematic diagram of intel 8086 registers

24. Describe in brief the four segment registers. [Model Question]

Answer:
The four segment registers are CS, DS, ES and SS – standing for code segment register, data segment register, extra segment register and stack segment register respectively. When a particular memory being read or write into, the corresponding memory address is determined by the content of one of these four segment registers in conjunction with their offset addresses.
The contents of these registers can be changed so that program may jump from one active code segment to another one.
The use of these segment registers will be more apparent in memory segmentation schemes.

25. Discuss A_{16}/S_3-A_{19}/S_6 signals of 8086. [Model Question]

Answer:
These are time multiplexed signals. During T_1, they represent $A_{19} - A_{16}$ address lines. During I/O operations, these lines remain low. During T_2-T_4, they carry status signals.
S_4 and S_3 (during T_2 to T_4) identify the segment register employed for 20-bit physical address generation.
Status signal S_5 (during T_2 to T_4) represents interrupts enable status. This is updated at the beginning of each clock cycle.
Status signal S_6 remains low during T_2 to T_4.

26. Discuss the Reset pin of 8086. [Model Question]

Answer:

Reset is an active high input signal and must be active for at least 4 CLK cycles to be accepted by 8086. This signal is internally synchronised and execution starts only after Reset returns to low value.

For proper initialisation, reset pulse must *not* be applied before of 'power on' of the circuit. During Reset state, all three buses are tristated and ALE and HLDA are driven low.

During resetting, all internal register contents are set to 0000 H, but CS is set to F000 H and IP to FFF0 H. Thus execution starts from physical address FFFF0 H. Thus EPROM in 8086 is interfaced so as to have the physical memory location forms FFFF0 H to FFFFF H, i.e., at the end of the map.

27. Discuss the Instruction Pointer (IP) of 8086. [Model Question]

Answer:

Functionally, IP plays the part of Program Counter (PC) in 8085. But the difference is that IP holds the offset of the next word of the instruction code instead of the actual address (as in PC).

IP along with CS (code segment) register content provide the 20-bit physical (or real) address needed to access the memory. Thus CS:IP denotes the value of the memory address of the next code (to be fetched from memory).

Content of IP gets incremented by 2 because each time a word of code is fetched from memory.

28. Indicate the data types that can be handled by 8086 µP. [Model Question]

Answer:

The types of data formats that can be handle by 8086 fall under the following categories:

- Unsigned of signed integer numbers – both byte-wide and word-wide.
- BCD numbers – both in packed or unpacked from.
- ASCII coded data. ASCII numbers are stored one number per byte.

29. Is direct memory to memory data transfer possible in 8086? [Model Question]

Answer:

No, 8086 not have provision for direct memory to memory data transfer.

For this to be implemented, AX is used as an intermediate stage of data. The source byte (from the memory) is moved into AX register with one instruction. The second instruction moves the content of AX into destination location (into another memory location). As example,

```
MOV   AH,   [SI]
MOV   [DI],  AH
```

Here, the first instruction moves the content of memory location, whose offset address remains in SI, into AH. The second ensures that the content of AH is moved into another memory location whose offset address is in DI.

Long Answer Type Questions

1. Describe the following addressing modes with suitable example.
i) Register addressing mode
ii) Indirect addressing mode
iii) Indexed addressing mode
iv) Base addressing mode
v) Immediate addressing mode [WBSCTE 2019, 2021]

Answer: *Refer to Question No. 6 of Long Answer Type Questions.*

2. Describe the Flag Register of 8086 microprocessor. [WBSCTE 2022]

Answer:

The 8086/8088 flag register is represented in figure below. The 8086 has a 16 bit flag register which contain 9 active flags. These 9 active flags can be divided into two types.

(a) **Conditional flag:** Conditional flag are set of reset on the basis of the result renerated from the ALU.

(b) **Control flag:** The 8086 contains three control flags which are used for controlling the microprocessor. These flags are set or reset with the help of instructions that are put in a program and are used to control program flow.

Specifically, we shall now first discuss the respective flags.

1. Carry Flag: This conditional flag is set when a carry is generated in the ALU as shown by the example given below.

```
e.g., ADD  F F F F h,      A A A A h
           F F F F
           AAAA
          (1)AAA 9
```

Carry

Thus carry flag = 1 [Carry flag is set to shown the carry generated]

In the above operation we fine that after two 16 bit numbers are added their sum is 17 bits wide. This excess bit which cannot be contained in a 16 bit register or memory location is used to set the carry flag thereby can be used by a programmer to check the state of a result.

2. Auxiliary Carry Flag: This flag is set when a carry is generated after the 4th bit when data being operated is 8 bit wide. When the data being operated is 16 bits wide the carry if generated after the 8 bit will set the Auxiliary Carry Flag (ACF).

e.g. carry 1 1 1 1 1 1 1 Carry after 4 bits (shown by the arrow)

Byte 1	1 1 1	1 1 1 1	Sets AF = 1.
+Byte 2	+0 0 0 0	0 0 0 1	However, if this carry is a
	1 0 0 0	0 0 0 0	

The auxiliary carry flag is used during arithmetic operations ion BCD number.

3. Zero Flag: This flag is set when the ALU generates a zero result. A zero would set the zero flag while a non-zero result will reset the zero flag.

4. Sign Flag: The sign flag is set when the MSB' of the result is one. This flag is used as during arithmetic operations on signed numbers.

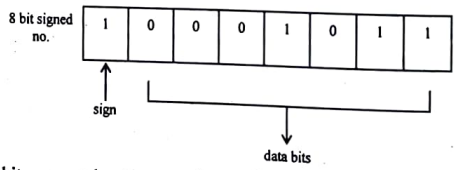
5. Overflow Flag: This flag is set when there is an overflow. For example when two number n bit are added and there is a carry generated in the (n-1) bits then an overflow is said to have occurred when the two n bit numbers are assumed to be signed numbers.

Carry [1] ← Overflow (carry in (n-1) bit)

[1] [1]

data	0 1 0 0 1 1 0 0 1
data 2	1 1 0 0 0 1 0 0 1
	1 0 0 0 1 0 0 0 0

This flag is used when two signed numbers (where the MSB bit is used as a sign bit) are added.



Here seven (7) bits are used as the data bits, thus a carry these 7 bits if generated would also be added to the MSB (the sign bit) it would result in wrong results. This carry is used to set the overflow flag sp that 1 programmer can check the overflow flag and thereby be informed that the result is invalid (wrong). A programmer can then initiate necessary step to correct the results.

6. Parity* Flag: Parity flag is set when the ALU output has even parity and reset when the ALU output has odd parity.

7. Trap Flag: Used for single stepping through a program**.

8. Interrupt Flag: When the interrupt flag is set a program can be interrupted for other more important operations. When it is reset the program executed cannot be interrupted. (Refer to Section on Interrupts)

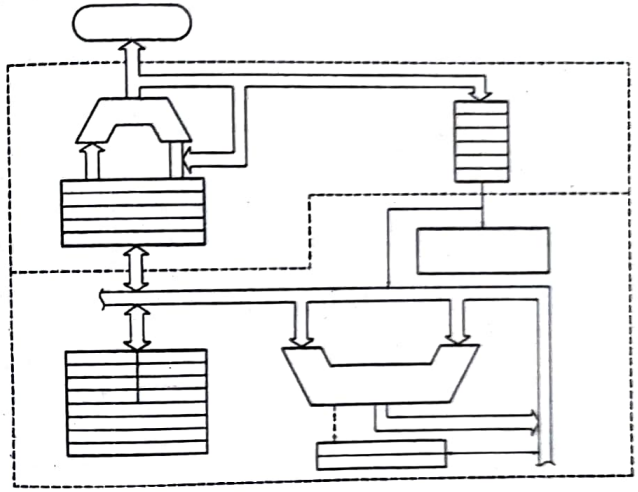
9. Direction Flag: This is used with string operation***.

Of the above flags, flags 1 through 6 are conditional flags while flags 7 to 9 are control flags.

➲ **3. Explain the architecture of 8086.** **[Model Question]**

Answer:

The following diagram depicts the architecture of a 8086 Microprocessor -



8086 Microprocessor is divided into two functional units, i.e., **EU** (Execution Unit) and **BIU** (Bus Interface Unit).

EU (Execution Unit)

Execution unit gives instruction to BIU Stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure; it performs operations over data through BIU.

Let us now discuss the functional parts of 8086 microprocessors.

ALU

It handles all arithmetic and logical operations, like +, -, ×, /, OR, AND, NOT operations.

Flag Register
It is a 16-bit register that behaves like a flip-flop, i.e. it changes accounting to result stored in the accumulator. It has 9 flags and they are divided into 2 groups – Conditional Flags and Control Flags.

Conditional Flags
It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags –

- **Carry flag** – This flag indicates an overview condition for arithmetic operations.
- **Auxiliary flag** – When an operation is performed at ALU, it results in a Carry/barrow from lower nibble (i.e. D0 - D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
- **Parity flag** – This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Sign flag** – This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- Overflow flag – This flag represents the result when the system capacity is exceeded.

Control Flags
Control flags controls the operations of the execution unit. Following is the list of control flags –

- **Trap flag** – It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** – It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enable condition and set to 0 for interrupt disabled condition.
- **Direction flag** – It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-verse.

General purpose register
There are 8 general purpose registers, i.e., AH, AL, BH, BL, GH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16 bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- **AX register** – It is also known as accumulator register. It is used to store operands for arithmetic operations.
- **BX register** – It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** – It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** – This register is used to hold I/O port address for I/O instruction.

Stack Pointer register
It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

BIU (Bus Interface Unit)
BIU takes care of all data and address transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.
It has the following functional parts –

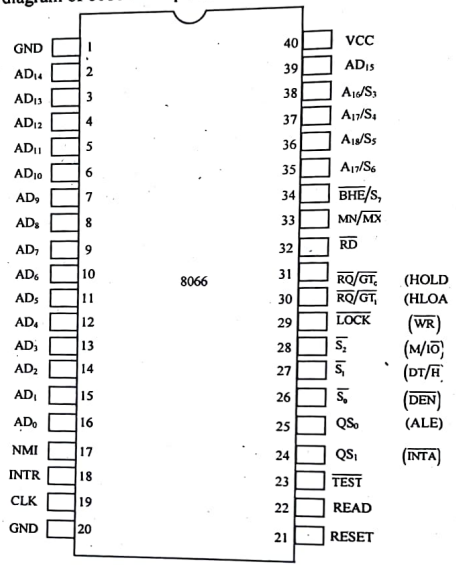
- **Instruction queue** - BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased executed speed.
- Fetching the next instruction while the current instruction executes is called **pipelining.**
- Segment register – BIU has 4 segment buses, i.e. CS, DS, SS & ES. It holds the addresses of instruction and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.
  - **CS** – It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.
  - **DS** – It stands for Data Segment. It consists of data used by the program and is accessed in the data segment by an offset address or the content of other register that holds the offset addresses.
  - **SS** – It stands for Stack Segment. It handles memory to store data and addresses during execution.
  - **ES** – It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.
- **Instruction Pointer** – It is a 16-bit register used to hold the address of the next instruction to be executed.

➲ 4. Describe 8086 pin diagram. **[Model Question]**

Answer:
8086 was the first 16-bit microprocessor available in 40-pin DIP (Dual Inline Package) chip. Let us now discuss in detail the pin configuration of a 8086 Microprocessor.

8086 Pin Diagram

Here is the pin diagram of 8086 microprocessor -



Let us now discuss the signals in details –

Power supply and frequency signals

It uses 5V DC supply at V_{cc} pin 40, and ground at V_{ss} pin 1 and 20 for its operation.

Clock signal

Clock signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different versions, i.e. 5 MHz, 8 MHz and 10 MHz.

Address/data bus

AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8AD16 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it

Address/status bus

A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and late it carries status signals.

S7/BHES

BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.

Read($\overline{RD}$)

It is available at pin 32 and is used to read signal for read operation.

Ready

It is available at pin 22. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

RESET

It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

INTR

It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

NMI

It stands for Non-maskable interrupts and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.

$\overline{TEST}$

This signal is like wait state and is available at pin 23. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.

MN/$\overline{MX}$

Its stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-a-versea.

INTA

It is an interrupt acknowledgement signal and id available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.

ALE

It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

DEN

It stands for Data Enable and is available at pin 26. It is used to enable Transreceiver 8086. The transreceiver is a device used to separate data from the address/data bus.

DT/R

It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the transreceiver. When it is high, data is transmitted out and vice-a-versa.

M/IO
This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicates the memory operation. It is available at pin 28.

WR
It stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.

HLDA
It stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.

HOLD
This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.

QS_1 and QS_0
These are queue status signals and are available at pin 24 and 25. These signals provide the status of instruction queue. Their conditions are shown in the following table-

QS_0	QS_1	Status
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

S_0, S_1, S_2
These are the status signals that provide the status of operation, which is used by the Bus Controller 8088 t0 generate memory & I/O control signals. These are available at pin 26, 27, and 28. Following is the table showing their status –

S_2	S_1	S_0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O White
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

LOCK
When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.

RQ/GT_1 and RQ/GT_0
These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgement. RQ/GT_0 has a higher priority than RQ/GT_1.

5. Describe 8086 instruction set. **[Model Question]**

Answer:
The 8086 microprocessor supports 8 types of instructions –

- Data transfer instructions
- Arithmetic Instructions
- Bit Manipulation Instructions
- String Instructions
- Program Execution Transfer instructions (Branch & Loop Instructions)
- Processor Control Instructions
- Iteration Control Instructions
- Interrupt Instructions

Let us now discuss these instruction sets in detail.

Data Transfer Instructions
These instructions are used to transfer the data from the source operand to the destination operand.
Following are the list of instructions under this group –

Instruction to transfer a word

- **MOV** – Used to copy the byte or word from the provided source to the provided destination.
- **PPUSH** – Used to put a word at the top of the stack.
- **POP** – Used to get a word from the top of the provided location.
- **PUSHA** – Used to pull all the registers into the stack.
- **POPA** – Used to get words from the stack to all registers.
- **XCHG** – Used to exchange the data from two locations.
- **XLAT** – Used to translate a byte in AL using a table in the memory.

Instructions for input and output port transfer

- **IN** – Used to read a byte or word from the provided port to the accumulator.
- **OUT** – Used to send out a byte or word from the accumulator to the provided port.

Instructions to transfer the address

- **LEA** – Used to load the address of operand into the provided register.
- **LDS** – Used to load DS register and other provided register from the memory
- **LES** – Used to load ES register and other provided register from the memory.

Instructions to transfer flag registers

- **LAHF** – Used to load AH with the low byte of the flag register.
- **SAHF** – Used to store AH register to low byte of the flag register.
- **PUSHF** – Used to copy the flag register at the top of the stack.

- **POPF** – Used to copy a word at the top of the stack to the stack.

Arithmetic Instructions

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc.

Following is the list instructions under this group –

Instructions to perform addition

- **ADD** – Used to add the provided byte to byte/word to word.
- **ADC** – Used to add with carry.
- **INC** – Used to increment the provided byte/word by 1.
- **AAA** – Used to adjust ASCII after addition.
- **DAA** – Used to adjust the decimal after the addition/subtraction operations.

Instructions to perform subtraction

- **SUB** – Used to subtract the byte from byte/word from word.
- **SBB** – Used to perform subtraction with borrow.
- **DEC** – Used to decrement the provided byte/word by 1.
- **NPG** – Used to negate each bit of the provided byte/word and add1/2's complement.
- **CMP** – Used to compare 2 provided byte/word.
- **AAS** – Used to adjust ASCII codes after subtraction.
- **DAS** – Used to adjust decimal after subtraction.

Instructions to perform multiplication

- **MUL** – Used to multiply unsigned byte by byte/word by word.
- **IMUL** – Used to multiply signed byte by byte/word by word.
- **AAM** – used to adjust ASCII codes after multiplication.

Instructions to perform division

- **DIV** – Used to divide the unsigned word by byte or unsigned double word by word.
- **IDIV** – Used to divide the signed word by byte or signed double word by word.
- **AAD** – Used to adjust ASCII codes after division.
- **CBW** – Used to fill the upper word of the double word with the sign bit of the lower word.

Bit Manipulation Instructions

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc.

Instructions to perform logical operation

- **NOT** – Used to invert each bit of a byte or word.
- **AND** – Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
- **OR** - Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
- **XOR** – Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.

- **TEST** – Used to add operands to update flags, without affecting operands.

Instructions to perform shift operations

- **SHL/SAL** – Used to shift bits of a byte/word towards left and put zero(S) in LSBs.
- **SHR** – Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
- **SAR** – Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB.

Instructions to perform rotate operations

- **ROL** – Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
- **ROR** – Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
- **RCR** – Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
- **RCL** - Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

String Instructions

String is a group of bytes/word and their memory is always allocated in a sequential order.

Following is the list of instructions under this group –

- **REP** – Used to repeat the given instruction till CX ≠ 0.
- **REPE/REPZ** – Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **REPNE/REPNZ** – Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.
- **MOVS/MOVSB/MOVSW** – Used to move the byte/word from one string to another.
- **COMS/COMPSB/COMPSW** - Used to compare two string bytes/words.
- **INS/INSB/INSW** – Used as an input string/byte/word from the I/O port to the provided memory location.
- **OUTS/OUTSB/OUTSW** – Used as an output string/byte/word from the provided memory location to the I/O port.
- **SCAS/SCASB/SCASW** – Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.
- **LODS/LODSB/LODSW** – Used to store the string byte into AL or string word into AX.

Program Execution Transfer Instructions (Branch and Loop Instructions)

These instructions are used to transfer/branch the instructions during an execution. It includes the following instructions –

Instructions to transfer the instruction during an execution without any condition –

- **CALL** – Used to call a procedure and save their return address to the stack.
- **RET** – Used to return from the procedure to the main program.

- **JMP** – Used to jump to the provided address to proceed to the next instruction.

Instructions to transfer the instruction during an execution with some condition –

- **JA/JNBE** – Used to jump if above/not below/equal instruction satisfies.
- **JAE-JNB** - Used to jump if above/not below instruction satisfies.
- **JBE/JNA** - Used to jump if below/equal/not above instruction satisfies.
- **JC** – Used to jump if flag CF = 1
- **JE/JZ** – Used to jump if equal/zero flag ZF = 1
- **JG/JNLE** – Used to jump if grater/not less than/equal instruction satisfies.
- **JGE/JNL** - Used to jump if grater/equal/not less than instruction satisfies.
- **JL/JNGE** - Used to jump if less than/not greater than/equal instruction satisfies.
- **JLE/JNG** - Used to jump if less than/equal/if not greater than instruction satisfies.
- **JNC** – Used to jump if no carry flag (CF = 0)
- **JNE/JNZ** – Used to jump if not equal/zero flag ZF = 0
- **JNO** – Used to jump if no overflow flag OF = 0
- **JNP/JPO** – Used to jump if not parity/parity odd PF = 0
- **JNS** – Used to jump if not sign SF = 0
- **JO** – Used to jump if overflow flag OF = 1
- **JP/JPE** – Used to jump if sign flag SF = 1

Processor Control Instructions

These instructions are used to control the processor action by setting/resetting the flag values.

Following are the instructions under thi8s group –

- **STC** – Used to set carry flag CF to 1
- **CLC** – Used to clear/reset carry flag CF to 0
- **CMC** – Used to put complement at the state of carry flag CF.
- **STD** – Used to set the direction flag DF to 1
- **CLD** – Used to clear/reset the direction flag DF to 0
- **STI** – Used to set the interrupt enable flag to 1, i.e., enable INTR input.
- **CLI** - Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

Iteration Control instructions

These instructions are used to execute the given instructions for number of times. Following is the list of instructions under this group –

- **LOOP** – Used to loop a group of instructions until condition satisfies, i.e., CX = 0
- **LOOPE/LOOPZ** – Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0
- **LOOPNE/LOOPNZ** – Used to loop a group of instructions till satisfies ZF = 0 & CX = 0
- **JCXZ** – Used to jump to the provided address if CX = 0

Interrupt Instructions

These instructions are used to call the interrupt during program execution.

- **INT** – Used to interrupt the program during execution and calling service specified.
- **INTO** – Used to interrupt the program during execution if OF = 1
- **IRET** – Used to return from interrupt service to the main program

6. What are the addressing mode of 8086? **[Model Question]**

Answer:

The way for which an operand is specified for an instruction in the accumulator, in a general purpose register or in memory location, is called **addressing mode.**

The 8086 microprocessors have 8 addressing modes. Two addressing modes have been provided for instructions which operate on register or immediate data.

These two addressing modes are:

Register Addressing: In register addressing, the operand is placed in one of the 16-bit or 8-bit general purpose registers.

Example

- MOV AX, CX
- ADD AL, BL
- ADD CX, DX

Immediate Addressing: In immediate addressing, the operand is specified in the instruction itself.

Example

- MOV AL, 35H
- MOV BX, 0301H
- MOV [0401], 3598H
- ADD AX, 4836H

The remaining 6 addressing modes specify the location of an operand which is placed in a memory.

These 6 addressing modes are:

Direct Addressing: In direct addressing mode, the operand?s offset is given in the instruction as an 8-bit or 16-bit displacement element.

Example

- ADD AL, [0301]

The instruction adds the content of the offset address 0301 to AL. the operand is placed at the given offset (0301) within the data segment DS.

Register Indirect Addressing: The operand's offset is placed in any one of the registers BX, BP, SI or DI as specified in the instruction.

Example

- MOV AX, [BX]

It moves the contents of memory locations addressed by the register BX to the register AX.

Based Addressing: The operand's offset is the sum of an 8-bit or 16-bit displacement and the contents of the base register BX or BP. BX is used as base register for data segment, and the BP is used as a base register for stack segment.

Effective address (Offset) = [BX + 8-bit or 16-bit displacement].

Example
- MOV AL, [BX+05]; an example of 8-bit displacement.
- MOV AL, [BX + 1346H]; example of 16-bit displacement.

Indexed Addressing: The offset of an operand is the sum of the content of an index register SI or DI and an 8-bit or 16-bit displacement.
Offset (Effective Address) = [SI or DI + 8-bit or 16-bit displacement]

Example
- MOV AX, [SI + 05]; 8-bit displacement.
- MOV AX, [SI + 1528H]; 16-bit displacement.

Based Indexed Addressing: The offset of operand is the sum of the content of a base register BX or BP and an index register SI or DI.
Effective Address (Offset) = [BX or BP] + [SI or DI]
Here, BX is used for a base register for data segment, and BP is used as a base register for stack segment.

Example
- ADD AX, [BX + SI]
- MOV CX, [BX + SI]

Based Indexed with Displacement: In this mode of addressing, the operand's offset is given by:
Effective Address (Offset) = [BX or BP] + [SI or DI] + 8-bit or 16-bit displacement

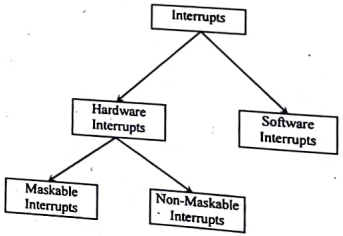
Example
- MOV AX, [BX + SI + 05]; 8-bit displacement
- MOV AX, [BX + SI + 1235H]; 16-bit displacement

➲ **7. Write the interrupt mechanism of 8086.** **[Model Question]**

Answer:

Interrupt is a process of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.
Microprocessor responds to these interrupts with an **interrupt service routine (ISR)**, which is a short program or subroutine to instruct the microprocessor on how to handle the interrupt.

There are different types of interrupt in 8086:



Hardware Interrupts

Hardware interrupts are that type of interrupt which are caused by any peripheral device by sending a signal through a specified pin to the microprocessor.

The Intel 8086 has two hardware interrupt pins:
- NMI (Non-Maskbale Interrupt)
- INTR (Interrupt Request) Maskable Interrupt.

NMI: NMI is a single Non-Maskable Interrupt having higher priority than the maskable interrupt.
- It cannot be disabled (masked) by user using software.
- It is used by the processor to handle emergency conditions.
  For example: It can be used to save program and data in case of power failure. An external electronic circuitry is used to detect power failure, and to send an interrupt signal to 8086 through NMI line.

INTR: The INTR is a maskable interrupt. It can be enabled/disabled using interrupt flag (IF). After receiving INTR from external device, the 8086 acknowledges through INTA signal.
It executes two consecutive interrupt acknowledge bus cycles.

Software Interrupt

A microprocessor can also be interrupted by internal abnormal conditions such as overflow; division by zero; etc. A programmer can also interrupt microprocessor by inserting INT instruction at the desired point in the program while debugging a program. Such an interrupt is called a software interrupt.
The interrupt caused by an internal abnormal conditions also came under the heading of software interrupt.

Example of software interrupts are:
- TYPE 0 (division by zero)
- TYPE 1 (single step execution for debugging a program)
- TYPE 2 represents NMI (power failure condition)
- TYPE 3 (break point interrupt)
- TYPE 4 (overflow interrupt)

Interrupt pointer table for 8086

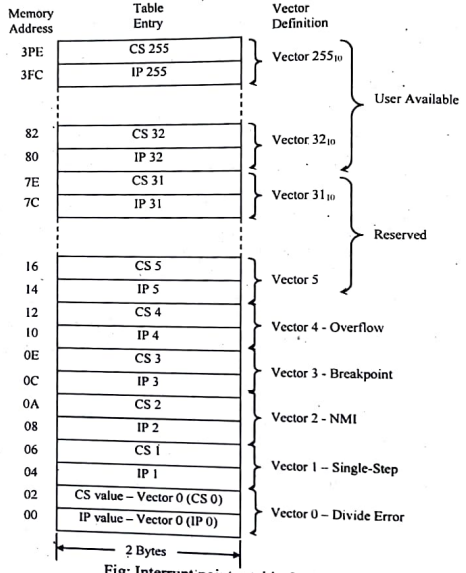


Fig: Interrupt pointer table for 8086

The 8086 can handle up to 256, hardware and software interrupts. 1KB memory acts as a table to contain interrupt vectors (or interrupt pointers), and it is called interrupt vector table or interrupt pointer table. The 256 interrupt pointers have been numbered from 0 to 255 (FF hex). The number assigned to an interrupt pointer is known as type of that interrupt. For example, Type 0, Type 1, Type 2,.....Type 255 interrupt.

➲ 8. Describe in detail, the general purpose of data register. **[Model Question]**

Answer:

All the four registers can be used bytewise or wordwise. The alphabets X, H and L respectively refer to word, higher byte or byte or lower byte respectively of ant register. All the four registers can be used as the source or destination of an operand during an arithmetic operation such as ADD or logical operation such as AND or logical operation such as AND, although particular registers are earmarked for specific operations.

Register C is used as a count register in string operations and as such is called a 'count' register. Register C is also used for multibit shift or rotate instructions.
Register D is used to hold the address of I/O port while register A is used for all I/O operations that require data to be inputted or outputted.

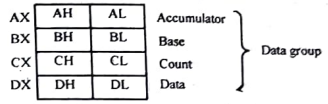


Figure below shows the four data registers along with their dedicated functions also.

Register	Operations
AX	Word multiply, Word divide, Word I/O
AL	Byte multiply, byte divide, byte I/O, translate, decimal arithmetic
AH	Byte multiply, byte divide
BX	Translate
CX	String operations, Loops
CL	Variable shift and rotate
DX	Word multiply, Word divide, indirect I/O

Fig: Data group registers and their functions

➲ 9. Describe the status register of 8086. **[Model Question]**

Answer:

It is a 16-bit register, also called flag register or Program Status Word (PSW). Seven bits remain unused while the reset nine are used to indicate the conditions of flags. The status flags of the register are shown below in figure below.

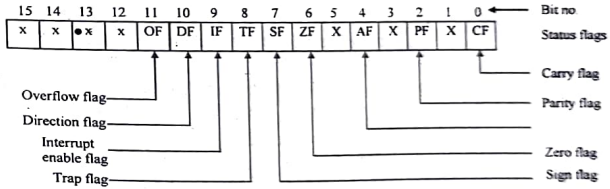


Fig: Status flags of intel 8086

Out of nine flags, six are condition flags and three are control flags. The control flags are TF (Trap), IF (Interrupt) and DF (Direction) flags, which can be set/reset by the programmer, while the condition flags [OF (Overflow), SF (Sign), ZF (Zero), AF (Auxiliary Carry), PF (Parity) and CF (Carry)] are set/reset depending on the results of some arithmetic or logical operations during program execution.
CF is set if there is a carry out of the MSB position resulting from an addition operation or if a borrow out of the MSB position during subtraction.

PF is set if the lower 8-bits of the result of an operation contains an even number of 1's.
AF is set if there is a carry out of bit 3 resulting from an addition operation or a borrow required from bit 4 into bit 3 during subtraction operation.
ZF is set if the result of an arithmetic or logical operation is zero.
SF is set if the MSB of the result of an operation is 1. SF is used with unsigned numbers.
OF is used only for signed arithmetic operation and is set if the result is too large to be fitted in the number of bits available to accommodate it.
The function of the flags along with their bit positions are shown in Fig. below.

Bit position	Name	Function
0	CF	Carry flag: Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity flag: Set if low-order 8-bit of result contain an even number of 1-bit; cleaned otherwise
4	AF	Set on carry from or borrow to the low-order 4-bits of AL; cleaned otherwise
6	ZF	Zero flag: Set if result is zero; cleaned otherwise
7	SF	Sign Flag: Set equal to high-order bit of result (0 is positive, 1 if negative)
8	TF	Signal step flag: Once set, a signal-step interrupt occurs after the next instruction executes; TF is cleaned by the single-step interrupt
9	IF	Interrupt-enable flag: When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified Location.
10	DF	Direction flag: Causes string instructions to auto decrement the appropriate index register when set; clearing DF causes auto increment.
11	OF	Overflow flag: Set if the signed result cannot be expressed within the number of bits in the destination operand; cleared otherwise.

Fig: 8086 flags: DF. IF and TF can be set or reset to control the operations of the processor. The remaining flags are status indicators.

➲ 10. Discuss the pointers and index group of registers. **[Model Question]**

Answer:

The pointer registers are SP and BP while the index registers are SI and DI.
All the four are 16-bit registers and are used to store offset addresses of memory locations relative to segment registers. They act as memory pointers. As an example, MOV AH, [SI] implies, "Move the byte whose address is contained in SI into AH". If now, SI = 2000 H, then execution of above instruction will Put the value FF H in register

AH, shown in Fig. 11.10, [SI+1 : SI] = ABFF H, where obviously SI+1 points to memory location 2001 H and [SI+1] = AB H.
SI and DI are also used as general purpose registers. Again in certain string (block move) instructions, SI and DI are used as source and destination index registers respectively. For such cases, contents of SI are added to contents of DS register to get the actual source address of data, while the contents of DI are added to the contents of ES to get the actual destination address of data.
SP and BP stand for stack pointer and base pointer with SP containing the offset address or the stack top address. The actual address is computed by adding the contents of SP and SS.
Data area(s) may exist in stack. To access such data are in stack segment, BP register is used which contains the offset address. BP register is also used as a general purpose register.
Instruction Pointer (IP) is also included in the index and pointers group. IP points to the offset instead of the actual address of the next instruction to be fetched (from the current code segment) in BIU. IP resides in BIU but cannot be programmed by the programmer.

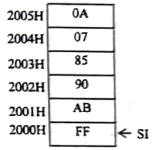


Fig: SI is pointing to memory locations 2000 H.

➲ 11. Discuss the instruction format of 8086. **[Model Question]**

Answer:

The instruction format of 8086 is shown in Fig. below. It is extendable up to 6-bytes. The first byte contains D and W – Direction Register Bit and Data Size Bit respectively. Both D and W are 1-bit in nature.

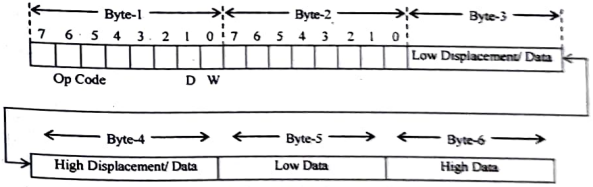


Fig: 8086 instruction format

- If D = 1, then register operand existing in byte-2 is the destination operand, otherwise (i.e., if D = 0) it is a source operand.
- W indicates whether the operation is and 8-bit or a 16-bit data. If W = 0 then it is an 8-bit operation, else (i.e., W = 1) it is 16-bit one.
- The 2nd byte (byte-2) indicates whether one of the operands is in memory or both are in registers. This bytes contains three fields:

Field	Abbreviation	Length (no. of bits)
Mode field	MOD	2
Register field	REG	3
Register/Memory fields	r/m	3

Unit 4: Assembly Language Programming

Unit at a glance:

4.1 What is an assembly language?

An assembly language is a type of low-level programming language that is intended to communicate directly with a computer's hardware. Unlike machine language, which consists of binary and hexadecimal characters, assembly languages are designed to be readable by humans.

Low-level programming languages such as assembly language are a necessary bridge between the underlying hardware of a computer and the higher-level programming languages—such as Python or JavaScript—in which modern software programs are written.

4.2 How Assembly Languages Work?

Fundamentally, the most basic instructions executed by a computer are binary codes, consisting of ones and zeros. Those codes are directly translated into the "on" and "off" states of the electricity moving through the computer's physical circuits. In essence, these simple codes form the basis of "machine language," the most fundamental variety of programming language.

Of course, no human would be able to construct modern software programs by explicitly programming ones and zeros. Instead, human programmers must rely on various layers of abstraction that can allow themselves to articulate their commands in a format that is more intuitive to humans.

Specifically, modern programmers issue commands in so-called "high-level languages," which utilize intuitive syntax such as whole English words and sentences, as well as logical operators such as "and," "or," and "else" that are familiar to everyday usage.

Ultimately, however, these high-level commands need to be translated into machine language. Rather than doing so manually, programmers rely on assembly languages whose purpose is to automatically translate between these high-level and low-level languages. The first assembly languages were developed in the 1940s, and though modern programmers and modern natural language processors spend very little time dealing with assembly languages, they nevertheless remain essential to the overall functioning of a computer.

4.3 Components of Assembly Language

Syntax: When writing any code in any program language, there is an observable, specific order of rules that must be followed to allow a compiler to execute the code without error. These rules are defined as the syntax, and they contain criteria such as the maximum number of allowable characters, what characters code lines must start with, or what certain symbols "i.e. a semi-colon" means.

Label: A label is a symbol that represents the address where an instruction or data is stored. It's purpose is to act as the destination when referenced in a statement. Labels can be used anywhere an address can be used in assembly languages. A symbolic label consists of an identifier followed by a colon, while numeric labels consist of a single digital followed by a colon.

Operators: Also referred to as commands, operators are logical expressions that occur after the label field. In addition, it must be preceded by at least one white-space character. Operators can either be opcode or directive. Opcode correspond directly to machine instructions, and the operation code includes any register name associated with the instruction. Alternatively, directive operation codes are instructions known by the assembler.

Directive: Directives are instructions to the assembler that tell what actions must take place during the assembly process. Directives have the importance of declaring or reserving memory for variables; these variables can be recalled later in processes to perform more dynamic functions. Directives are also used to break programs into different sections.

Macro: An assembly language macro is a template shoe format presents a series or pattern of statements. This sequence of assembly language statements might be common to multiple different programs. A macro facility is used to interpret macro definitions, while a macro call is inserted into the source code where "normal" assembly code would have gone instead of the macro set of statements.

Mnemonic: A mnemonic is an abbreviation for an operation. A mnemonic is entered into the operation code for each assembled program instruction to specify a shortened "opcode" that represents a larger, complete set of codes. For example, the mnemonic "multiply by two" has a full set of code that carries out the mnemonic.

4.4 Structure of Assembly Language

An assembly language program is a series of statements, which are either assembly language instructions such as ADD and MOV, or statements called **directives**.

An **instruction** tells the CPU what to do, while a **directive** (also called **pseudo-instructions**) gives instruction to the assembler. For example, ADD and MOV instructions are commands which the CPU runs, while ORG and END are assembler directives. The assembler places the opcode to the memory location 0 when the ORG directive is used, while END indicates to the end of the source code. A program language instruction consists of the following four fields –

[label:] mnemonics [operands] [;comment]

A square bracket ([ ]) indicates that the field is optional.

- The **label field** allows the program to refer to a line of code by name. The label fields cannot exceed a certain number of characters.
- The **mnemonics** and **operands fields** together perform the real work of the program and accomplish the tasks. Statements like ADD A , C & MOV C, #68 where ADD and MOV are the mnemonics, which produce opcodes ; "A, C" and "C, #68" are operands. These two fields could contain directives. Directives do

not generate machine code and are used only by the assembler, whereas instructions are translated into machine code for the CPU to execute.

```
1. 0000          ORG   0H            ; start (origin) at location 0
2. 0000  7D25    MOV   R5, 25H       ; load 25H into R5
3. 0002  7F34    MOV   R7, 34H       ; load 34H into R7
4. 0004  7400    MOV   A,   #0       ; load 0 into A
5. 0006  2D      ADD   A,   R5       ; add contents of R5 to A
6. 0007  2F      ADD   A,   R7       ; add contents of R7 to A
7. 0008  2412    ADD   A,   #12H     ; add to A value 12 H
8. 000A  80FE    HERE:  SJMP HERE    ; stay in this loop
9. 000C  END                         ; end of asm source file
```

- The **comment field** begins with a semicolon which is a comment indicator.
- Notice the Label "HERE" in the program. Any label which refers to an instruction should be followed by a colon.

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. The alternate way of writing the instruction, ADD #5, R1 is **[WBSCTE 2022]**

(a) ADD [5], [R1] (b) ADDI 5, R1

(c) ADDIME 5, [R1] (d) There is no other way

Answer: (b)

2. The instructions that are used for reading an input port and writing an output port respectively are **[WBSCTE 2022]**

(a) MOV, XCHG (b) MOV, IN (c) IN, MOV (d) IN, OUT

Answer: (d)

3. The advantages of assembly level programming are **[Model Question]**

(a) flexibility of programming is more (b) chances of error are less

(c) debugging is easy (d) all of the mentioned

Answer: (d)

4. Assembly language programs are written using **[Model Question]**

(a) Hex coda (b) Mnenonics

(c) ASCII code (d) None of these View

Answer: (b)

B. Fill in the blanks in the following statements:

5. MOV AX. [2A50] is an example of ***direct*** addressing mode. [WBSCTE 2022]

C. Answer the following questions:

6. What is Assembler? [Model Question]

Answer:

An assembler can be defined as a program, which accepts a symbolic language program called the source program as the input and produces the binary language equivalent of the input (i.e. the result is a binary program), which is called the object program.

7. What are the advantages of assembly language? [Model Question]

Answer:

1. **Faster:** Basically assembly language program are executed in much less time as compared to the high-level programming like C, C+.
2. **Low memory usage:** As assembly is processor specific it consumes less memory and are compiled in low memory space.
3. **Real Time System:** Real time applications use assembly because they have a deadline for their output. (i.e., system should response or generate output within a specific period of time.)

8. What are the disadvantages of assembly language? [Model Question]

Answer:

1. **Portability:** Assembly language is processor specific so it cannot run on multiple platforms. It is machine specific language.
2. **Difficult to program:** The programmer should have a keen knowledge about the architecture of the processor as different processors will have different register set and different combinations to use them.
3. **Debugging:** Debugging becomes very difficult for assembly language if program has some error.

9. Why to use assembly language programming? [Model Question]

Answer:

If you programming for a specific processor or for real time applications assembly language programming can be more useful to you in terms of processing speed, performance and in low memory system.

Long Answer Type Questions

➲ **1. Write an assembly language program to add two 8-bit numbers.**
OR,
Write an Assembly language program to add two numbers.
OR,
Write an Assembly level program to add two single byte numbers. [Model Question]

Answer:

Program

Label	Opcode	Operand	Comment
	LXI	H,2000 H	Load 16-bit address in HL.
	LDA	2001 H	Load accumulator with data of memory 2001.
	MOV	B, M	Copy data from memory pointed by HL into B.
	ADD	B	Add contents of B with contents of accumulator.
	STA	2002 H	Store the contents of accumulator in the memory location 2002.
	HLT		Stop processing.

➲ **2. Explain the first pass of an Assembler.** [Model Question]

Answer:

During the first pass of an assembler, the following sequences of events occur:

- The assembler checks whether all the instructions to be executed are correct as per the current assembly mode.
- It allocates adequate spaces for the instructions and results storage (or rather storage areas).
- Constant values are filled in as and where possible.
- The assembler then builds a symbol table, called the cross-reference table. This table then contains the entries for every symbol the assembler encounters in the label field of a given statement. So, these symbols stored in the table are used to identify the particular statements afterwards.

➲ **3. Write an Assembly – Language – Program to add a numbers where the numbers are stored in n consecutive locations (NUM, NUM+1....NUM +n – 1) and to store the result in memory location SUM. The number 'n' is stored in memory location N.** [Model Question]

Answer:

```
LDA XXXX // Some memory location say 8500
// Load the accumulator with the address of memory viz 8500
MOV D, A
// Move the accumulator value to the register D
```

```
MVI E, 07
// Load the E register with the counter 8 - 1
LXI B, XXXX + 1 // Next memory location from where we have done
LDA 8501
 // Load immediate B C with the memory location
  LDAX B
// Load Accumulator indirect A= [BC]
 ADD D
//Add the content of the Accumulator to the Register D
 MOV D, A
//Move the value in D to the Accumulator or register A
INX B
// Increment BC register pairs
DCR E
// Decrement the counter register
 JNZ XXXX // LDAX B Location
// Jump on no zero
STA XXXX // Suitable memory location
// Store the output into a memory location
HLT
// Stop the program
```

➲ 4. Write an assembly language program to subtract two 8-bit numbers. **[Model Question]**

Answer:

Program: Subtract two 8-bit numbers

Sample problem:

(4000H) = 51H
(4001H) = 19H

Result = 51H – 19H = 38H

Source program:

```
LXI H, 4000H      : "HL points 4000H"
MOV A, M          : "Get first operand"
INX H             : "HL points 4001H"
SUB M             : "Subtract second operand"
INX H             : "HL points 4002H"
MOV M, A          : "Store result at 4002H"
HLT               : "Terminate program execution"
```

➲ 5. What are basic assembler directives used in programming? **[Model Question]**

Answer:

1) ASSUME

Assume CS: CODE, DS: DATA

It is used to inform the complier that **CS (CODE SEGMENT)** contains the **CODE** and **DS (DATA SEGMENT)** contains **DATA**

Assume CS: DATA, DS: CODE

Here CODE is written in DATA SEGMENT and DATA in CODE SEGMENT

2) DUP()

Declaring an array with garbage

Eg. A DB 04H DUP (?)

A = Variable
DB = Data Type
04H = Length of Array
? = Element to be DUPLICATED (DUP)

Declaring an array with Same value

Eg. A DB 04H DUP (33H)

Defines the array with variable name A of length 04H having values 33H.

FOUR locations of array are having value 33H

Declaring an array with Different Elements

Eg. 1) A DB 03H, 04H, 05H

Eg. 2) A DB 'R', 'A', 'H', 'U', 'L

3) START

It indicates the start of Program.

4) END

It indicates the end of Program.

5) ENDS

Indicates End of Segment.

6) PROC

Used to indicate the begining of Procedure.

7) ENDP

Used to indicate the end of Procedure.

8) EQU

EQU (Equates) it is used for declaring variables having constants values.

Eg. A EQU 13H

Variable A is a constant having value 13H

➲ 6. Write about software interrupt. **[Model Question]**

Answer:

1) INT 03H

INT 03H (3) Breakpoint

INT 3 is the breakpoint interrupt.

Debuggers use this interrupt to establish breakpoints in a program that is being debugged. This is normally done by substituting an **INT 3** instruction, which is one byte long, for a byte in the actual program. The original byte from the program is restored by the debugged after it receives control through INT 3

2) KEYBOARD INTERRUPTS

Taking input from USER

i) MOV AH, 0AH
INT 21H

Keeps on taking input from user until terminated by '**$**'.

The input is taken in reg. AL

ii) MOV AH,01H
INT 21H
Takes only one character from user.
The input it taken in reg. AL

Display Massages
i) MOV AH, 09H
INT 21H
Displays a massage terminated by '$'.
The characters are taken in DX reg. (for word) or DL reg. (for byte) and Displayed.
ii) MOV AH, 02H
INT 21H
Displays only single Character whose ASCII value is in DL reg.

3) INT 10H

INT 10h / AH = 0 – set **video mode.**

Input:
AL = desired video mode.
These vide modes are supported:
00h – text mode. 40×25. 16 colors. 8 pages
03h - text mode. 80×25. 16 colors. 8 pages
13 h – graphical mode. 40×25. 256 colors. 320×200 plxels. 1 page.
Example: MOV AL, 13H
MOV AH, 0
INT 10H
***NOTE:** This Interrupt is used for clearing the DOS screen.

➲ 7. Explain LEA (LOAD EFFECTIVE (OFFSET) ADDRESS) instruction. **[Model Question]**

Answer:
The **LEA** instruction
LOAD EFFECTIVE (OFFSET) **ADDRESS**
LEA SI, A ; loads effective address of A in
; SI reg.
The above instruction can also be written as
MOV SI, OFFSET A
Eg. **A DB 01H,20H,30H,40H,50H**
To load the effective address of 50H in SI:
LEA SI, A+04H
This is because by Default **LEA SI, A** points at location **01H** to make it point at location **50H** we add **+04H**

➲ 8. Write an assembly language code to add two 16 bit nos. **[Model Question]**

Answer:

```
ASSUME CS: CODE, DS:DATA
DATA SEGMENT
   A DW 9384H
   B DW 1845H
   SUM DW?
   CARRY DB 00H
DATA ENDS
CODE SEGMENT
START:  MOV AX, DATA
   MOVE DS, AX
   MOVE AS, A
   JNC SKIP
   INC CARRY
   SKIP:  MOVE SUM, AX
   INT 03H
   CODE ENDS
   END START
```

➲ 9. Write an assembly language code to sort the nos in ascending order. **[Model Question]**

Answer:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
      A DB 0FFH,70H,90H,60H,0FEH,20H,10H,13H,25H,00H
DATA ENDS
CODE SEGMENT
    START :MOV AX,DATA
       MOV DS,AX
       MOV CX,0009H
    BACK:   MOV DX,0009H
       LEA SI,A
    BACK1:    MOV AL,[SI]
       INC SI
       CMP AL,[SI]
       JC SKIP
       XCHG AL,[SI]
       DEC SI
       MOV [SI],AL
       INC SI
    SKIP:     DEX DX
       JNZ BACK1
       LOOP BACK
       INT 03H
    CODE ENDS
    END START
```

10. Write an assembly language code to find largest of 10 nos. [Model Question]

Answer:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   A DB 10H,50H,40H,20H,80H,00H,00FFH,30H,60H,00FEH
DATA ENDS
CODE SEGMMENT
   START:   MOV AX,DATA
      MOV DS,AX
      LEA SI,A
      MOV BH,00H
   BACK:    CMP BH,[SI]
      JNC SKIP
      MOV BH,[SI]
   SKIP:    INC SI
      LOOP BACK
      MOV [SI],BH
      INT 03H
CODE ENDS
END START
```

11. To find the no of even & odd nos. from series of 10 nos. [Model Question]

Answer:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
   A DB 10H,15H,25H,16H,17H,19H,23H,77H,47H,34H
DATA ENDS
CODE SEGMENT
   START:   MOV AX,DATA
      MOV DS,AX
      LEA SI,A
      MOV BX,0000H
      MOV CX,000AH
   BACK:    MOV AL,[SI]
      ROR AL,I
      JC ODD
      INC BL
      JMP NEXT
   ODD:     INC BH
   NEXT:    INC SI
      LOOP BACK
      INT 03H
CODE ENDS
END START
```


12. To take String from user find its length and reverse the string. [Model Question]

Answer:

```
ASSUME DS: DATA,CS:CODE
DATA SEGMENT
   CR EQU 13D      ;EQU defines constant, CR and LF are constants
   LF EQU 10D      ;CARRIAGE RETURN and LINE FEED initialize with
            ; ASCII VALUES
   ER DB CR,LF'NO STRING ENTERED PRESS ANY KEY TO EXIT.....$'
   LEN DB CR,LF,'THE LENGTH OF STRING IS->$'
   REV DB CR,LF,'REVERSE OF YOUR SRTING->$'
   TEMP DB 00FFH DUP (?)
DATA ENDS
CODE SEGMENT
   START:   MOV AX,DATA ;Initialize DATA SEGMENT
      MOV DS,AX
      MOV AL,03H  ;CLEAR the DOS SEREEN
      MOV AH,0
      INT 10H
      MOV CX,0000H ;CLEAR the COUNT reg.
      MOV DX,OFFSET INPUT     ;Print the INPUT message
      MOV AH,09H
      INT 21H
      LEA DI,TEMP ;CHECKING whether STRING IS
      MOV AH,01H  ;PROVIDED
      MOV [DI],AL
      INC CX
      INC DI
      INT 21H
      CMP AL,13D
      JE EXIT
   BACK:    MOV AH,01H  ;KEEP ON taking CHARACTERS
      MOV [D1],AL ;until press ENTER
      INT 21H
      INC DI
      INC CX
      CMP AL,13D
      JNZ BACK
   MOV AH,09H     ;Print the LEN message
      LEA DX,LEN
      INT 21H
      DEC CL
      CMP CL,64H  ;CHECK for STRING LENGTH grater
            ;than 100D (64H)
```

```
PUSHF    ;CLEAR the OVERFLOW flag
   POP BX
   AND BH,00F7H
   PUSH BX
   POPF
3GE PRINT1
   MOV BX,CX
   CMP CL,OAH  ;CHECK for STRING LENGTH grater
   JGE SKIP   ;THE 10D (0AH)
   MOV BX,CX
   ADD BL,30H
   MOV AH,02H    ;PRINT the LENGTH for SINGLE
   MOVE DL,BL      ;DIGIT (FROM 1-9)
   INT 21H
   JMP SKIP1
   PRINT1:MOV AH,02H ;PRINT 1 AS MSB when length is grater
          ;than 99D
   MOV DL,31H
   INT 21H
SKIP:    MOV BL,CL   ;CONVERT the COUNT in BCD format
          ;for 2-DIGHT
   MOV AL,00H  ;COUNT
BACK0:   ADD AL,01H
   DAA
   DEC BL
   JNZ BACK0
   MOV BL,AL
   ROLL AL,01H ;MASK the LOWER NIBBLE & PRINT
   ROLL AL,01H
   ROLL AL,01H
   ROLL AL,01H
   AND AL,0FH
   ADD AL,30H
   MOV AH,02H
   MOV DL,AL
   INT 21H
AND BL,0FH     ;MASK the UPPER NIBBLE & PRINT
   ADD BL,30H
   MOV AH,02H
   MOV DL,BL
   INT 21H
SKIP1:   MOV AH;09H  ;Print the REV message
   MOV DX,OFFSET REV
   INT 21H
   MOV DI,OFFSET TEMP
                        ;Print the REVERSE STRING
```



```
   MOV BX,CX
   MOV AH,02H
BACK1:   MOV DL,[BX+DI]
   INT 21H
   DEC BX
   JNZ BACK1
   JMP LAST
EXIT:    MOV AH,09H  ;PRINT the ERROR message
          ; when no string is given
   LEA DX,ER
   INT 21H
LAST:    MOV AH,01H  ;HOLD THE O/P SCREEN
   INT 21H
   INT 03H
CODE ENDS
END START
```

13. Write short note on Single-pass assembler. **[Model Question]**

Answer:

Assembler translates the symbolic version of instructions into its binary version.

During the first pass of an assembler, the following sequences of events occur:

- The assembler checks whether all the instructions to be executed are correct as per the current assembly mode.
- It allocates adequate spaces for the instructions and results storage (or rather storage areas).
- Constant values are filled in as and where possible.
- The assembler then builds a symbol table, called the cross-reference table. This table then contains the entries for every symbol the assembler encounters in the label field of a given statement. So, these symbols stored in the table are used to identify the particular statements afterwards.

14. Write in short about macro and procedure. **[Model Question]**

Answer:

MACRO

Definition of the macro

A macro is a group of repetitive instructions in a program which are coded only once and can be used as many times as necessary.

The main difference between a macro and a procedure is that in the macro the passage of parameters is possible and in the procedure it is not, this is only applicable for the TASM – there are other programming languages which do allow it. At the moment the macro is

executed each parameter is substituted by the name or value specified at the time of the call.

Syntax of a Macro

The parts which make a macro are:

i) Declaration of the macro.
ii) Code of the macro
iii) Macro termination directive

The declaration of the macro is done the following way:
NameMacro MACRO [parameter1, parameter2...]

Eg. **To Display a massage**

```
DSPLY MACRO MSG
    MOV AH,09H
    LEA DX,MSG
    INT 21H
ENDM
```

To use a macro it is only necessary to call it by its name, as if it were another assembler instruction, since directives are no longer necessary as in the case of the procedures.

Example: **DLSPY MSG1**

PROC

Procedure

Definition of procedure

A procedure is a collection of instructions to which we can direct to flow of our program, and once the execution of these instructions is over control is given back to the next line to process of the code which called on the procedure.

At the time of invoking a procedure the address of the next instruction of the program is kept on the stack so that, once the flow of the program has been transferred and the procedure is done, one can return to the next line. Of the original program, the one which called the procedure.

Syntax of a Procedure

There are two types of procedures, the **INTRA-SEGMENTS**, which are found on the same segment of instructions, and the **INTRA-SEGMENTS** which can be stored on different memory segments.

When the intra-segment procedures are used, the value of IP is stored on the stack and when the intra-segments are used the value of CS:IP is stored.

The part which make a procedure are:

i) Declaration of the procedure
ii) Code of the procedure
iii) Return directive
iv) Termination of the procedure

Eg.

```
ADD PROC NEAR
MOV AX,30H
MOV BX,30H
ADD AX,BH
RET
ADD ENDP
```

To divert the flow of a procedure (calling it), the following directive is used:
CALL Name of the Procedure, Example

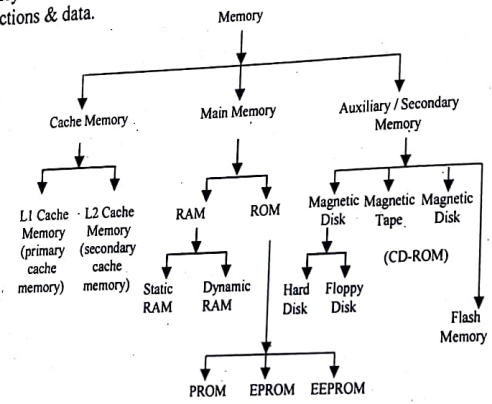
CALL ADD

Unit 5: Memory and Digital Interfacing

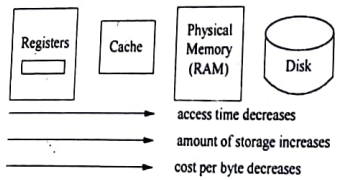
Unit at a glance:

5.1 Memory Classifications and Hierarchy

Memory is the storage unit of a computer i.e., the purpose of memory is to store both instructions & data.

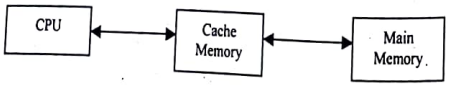


Memory Hierarchy



5.2 Cache memory, Mapping technique, Hit ratio, Replacement algorithm

Cache memory is a very small but very fast memory. It lies between the CPU and the main memory unit.



There are three popular methods of mapping addresses to cache locations.

Fully Associative – Search the entire cache for an address.
Direct – Each address has a specific each address has a specific place in the cache.
Set Associative– Each address can be in any of a small set of cache locations.
A cache-hit ratio is the number of times the database found something in cache divided by the number of times it looked for some object in the cache. The higher the ratio, the more effective the cache is at improving performance.

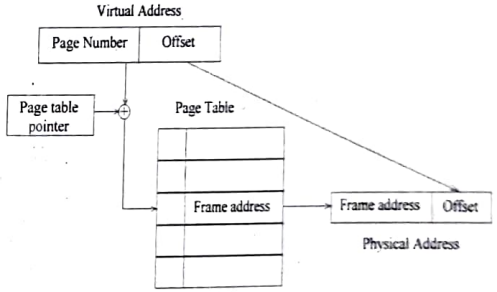
Some popular cache replacement algorithm

i) Belady's algorithm ii) Least recently used (LRU) algorithm iii) Most recently used (MRU) algorithm

5.3 Concept of virtual memory technique, address translation method, TLB

Computer hardware designers want to keep main memory as small as possible to keep prices down and make addressing simpler. Computer software designers want to have large programs that do many things, even though these programs may be too large to fit into main memory. The solution is virtual memory. The large program is divided up into "pages" which are brought into memory as needed.

To convert a virtual address into a physical address, the CPU uses the page number as an index into the page table. If the page is resident, the physical frame address in the page table is concatenated in front of the offset to create the physical address.



Translated Lookaside Table

When a new page is loaded into main memory, it usually writes over the page that has not been referenced for the longest time. This is not always the best idea. If the oldest page in memory has been referenced many times before and the other old pages have only been referenced once, you might be swapping out a page that is sure to be needed again instead of one that might not be needed again. Paging should be minimized or you can get to a place where you are spending more time swapping pages than you spend doing any actual work. This condition is called "thrashing" and should be avoided if you want the computer to run at top speed. One way to do this is to use translation lookaside tables to record more information about the history of pages so the computer can make better paging decision.

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. Which of the following comment about the program counter is true? [WBSCTE 2011]
   (a) It is used to count the no of instructions
   (b) It is a cell in ROM
   (c) During execution of the current instruction, its content changes
   (d) none of the above
Answer: (c)

2. The speed imbalance between memory access & CPU operation can be reduced by – [WBSCTE 2011, 2015, 2019]
   (a) Cache memory (b) memory interleaving
   (c) reducing memory size (d) none of the above
Answer: (b)

3. The cost of storing a bit is minimum in – [WBSCTE 2011, 2014, 2015, 2019]
   (a) Cache memory (b) Register (c) RAM (d) Tape
Answer: (d)

4. Which of the following statement(s) is true? [WBSCTE 2011, 2014, 2015, 2019]
   (a) ROM is a read/write memory
   (b) PC points to the last instruction that was executed
   (c) Stack works on the principle of LIFO
   (d) RAM is a Read/Write memory
Answer: (c) & (d)

5. Tera is 2 to the power of – [WBSCTE 2011, 2014, 2019]
   (a) 9 (b) 20 (c) 30 (d) 40
Answer: (d)

6. The difference between memory and storage is that memory is _____ and storage is _____ [WBSCTE 2011]
   (a) Temporary, permanent (b) Permanent, temporary
   (c) Slow, fast (d) All of above
Answer: (a)

7. Floating point representation is used to store – [WBSCTE 2012]
   (a) Boolean values (b) whole numbers
   (c) real integers (d) integers
Answer: (c)

8. The size gap between main memory & Secondary memory can be reduced by – [WBSCTE 2013]
   (a) Cache memory (b) memory interleaving
   (c) reducing memory size (d) virtual memory
Answer: (d)

9. Dirty bit can be used to represent – [WBSCTE 2013]
   (a) Noise in data
   (b) that the block has been modified in cache
   (c) that the memory write operation had been failed
   (d) none of the above
Answer: (c)

10. Cost per bit is maximum for – [WBSCTE 2013]
   (a) Cache memory (b) ROM (c) RAM (d) CPU Register
Answer: (c)

11. The access time is maximum in – [WBSCTE 2013]
   (a) Cache memory (b) Register (c) RAM (d) Hard disk drive
Answer: (d)

12. Giga is 2 to the power of – [WBSCTE 2013, 2015]
   (a) 9 (b) 20 (c) 30 (d) 40
Answer: (b)

13. The speed imbalance between memory access & CPU operation can be reduced by [WBSCTE 2014]
   (a) cache memory (b) memory interleaving
   (c) reducing memory size (d) none of these
Answer: both (a) and (b)

14. TLB is associated with – [WBSCTE 2015]
   (a) Cache memory (b) Virtual memory
   (c) Primary memory (d) All of the above
Answer: (a)

15. As per CPU concern Hard disk drive is – [WBSCTE 2015]
   (a) type of Memory (b) peripheral device
   (c) both (a) & (b) (d) none of the above
Answer: (b)

16. Which of the following statement is true about cache memory? – [WBSCTE 2016]
(a) It is used to increase the memory size
(b) It is used to decrease the memory size
(c) It decreases the overall performance of the CPU
(d) It minimizes the speed gap between CPU and main memory
Answer: (d)

17. MAR stands for – [WBSCTE 2016]
(a) Memory Access Register (b) Memory Address Register
(c) Memory Access Reference (d) Memory Address Reference
Answer: (b)

18. Which one is fastest? – [WBSCTE 2016]
(a) Cache memory (b) CPU Registers
(c) RAM (d) Hard disk drive
Answer: (b)

19. Cache memory uses – [WBSCTE 2016]
(a) SRAM (b) DRAM (c) EEPROM (d) EPROM
Answer: (a)

20. Replacement algorithm is not necessary is – [WBSCTE 2016]
(a) Associative mapping (b) Set associative mapping
(c) Direct mapping (d) All of the above
Answer: (c)

21. The three main components of a digital computer system are [WBSCTE 2017]
(a) Memory, IO, DMA (b) ALU, CPU, Memory
(c) CU, ALU, Register (d) CPU, Memory, IO
Answer: (d)

22. k-way set associative means [WBSCTE 2017, 2021]
(a) k blocks are present in a set (b) k sets are present in a block
(c) k sets are present in the cache (d) none of these
Answer: (a)

23. Virtual memory is [WBSCTE 2018]
(a) primary memory (b) secondary memory
(c) both a & b (d) none of these
Answer: (d)

24. Direct mapping is [WBSCTE 2018]
(a) one to one mapping (b) many to many mapping
(c) many to one mapping (d) all of these
Answer: (a)

25. In associative mapping technique, number of comparators required is [WBSCTE 2018]
(a) 1
(b) 2
(c) equal to the number of blocks in main memory
(d) equal to the number of lines in cache memory
Answer: (d)

26. The number successful accesses to memory stated as a fraction is called as_____. [WBSCTE 2022]
(a) Access rate (b) Success rate (c) Hit rate (d) Miss rate
Answer: (c)

27. In order to read multiple bytes of a row at the same time, we make use of [WBSCTE 2022]
(a) Memory extension (b) Cache
(c) Shift register (d) Latch
Answer: (a)

B. Fill in the blanks in the following statements:

28. EEPROM stands for ***Electrically Erasable Programmable Read-only Memory***. [WBSCTE 2010]

29. One sector of floppy disk contains ***512*** bytes of data. [WBSCTE 2010]

30. ***Hard Disk*** is an example of secondary memory. [WBSCTE 2010]

31. Digitizer is an ***Input*** device. [WBSCTE 2010]

32. ***Secondary memory*** is used to store data, instructions are results permanently for future use. [WBSCTE 2022]

33. ***Virtual memory*** is generally used to increase the apparent size of physical memory. [WBSCTE 2022]

34. A high speed memory is placed between the CPU and the primary memory is known as ***cache memory***. [WBSCTE 2022]

35. Technique that automatically move programs and data blocks into the physical memory when they are required for execution are called ***virtual memory technique***. [WBSCTE 2022]

36. Hit ratio is maximum in ***set associative*** mapping. [WBSCTE 2022]

37. Page table resides in ***main memory***. [WBSCTE 2022]

38. The smallest entity of memory is called ***cell***. [WBSCTE 2022]

C. Answer the following questions:

39. What is Hit Ratio? [WBSCTE 2013, 2016]
Answer: ***Refer to Article No. 5.2 of Unit at a glance.***

40. What is write back policy? [WBSCTE 2014]
Answer:
This caching technique improves the subsystem's response time to write requests by allowing the controller to declare the write operation 'complete' as soon as the data reaches its cache memory. The controller performs the slower operation of writing the data to the disk drives at a later time.

41. What is TLB? [WBSCTE 2014, 2019]
Answer: ***Refer to Article No. 5.3 of Unit at a glance.***

42. What do you mean by Replacement Algorithm? [WBSCTE 2014, 2019]
Answer:
The method used to determine which entry in an associative cache to flush to main memory when it is desired to cache a new block of data.
The "least recently used" algorithm flushed the block which has not been accessed for the longest time. A random replacement algorithm picks any block with equal probability.

43. What is paging?
Answer: [WBSCTE 2016]
Paging is a method of writing data to, and reading it from, secondary storage for use in primary storage, also known as main memory. Paging plays a role in memory management for a computer's OS (operating system).

44. What is the function of TLB?
Answer: ***Refer to Article No. 5.3 of Unit at a glance.*** [WBSCTE 2016]

45. What is L2 cache?
Answer: [WBSCTE 2018]
L2 (level 2) cache memory lies external to the processor chip i.e. L2 cache lies in-between the main memory and the processor chip. L2 cache is generally implemented using Static RAMs.

46. What is write through policy?
Answer: [WBSCTE 2019]
Write through is a storage method in which data is written into the cache and the corresponding main memory location at the same time. The cached data allows for fast

retrieval on demand, while the same data in main memory ensures that nothing will get lost if a crash, power failure, or other system disruption occurs.

47. What do you mean by Loop Buffer? [WBSCTE 2019]
Answer:
A loop buffer is a high-speed cache like memory present within the processor. It caches some of the previously executed instructions in sequence; the number of instructions cached depends upon the size of the loop buffer. The loop buffer differs from instruction cache in the following respect. While a cache memory can accommodate an arbitrary subset of the virtual address, the loop buffer strictly stores the last few instructions executed in sequence.

48. What do mean by vectored interrupt? [WBSCTE 2019]
Answer:
In computer science, a vectored interrupt is a processing technique in which the interrupting device directs the processor to the appropriate interrupt service routine.

49. What is the function of IO processor? [WBSCTE 2019]
Answer:
The Input Output Processor is a specialized processor which loads and stores data into memory along with the execution of I/O instructions. It acts as an interface between system and devices. It involves a sequence of events to executing I/O operations and then store the results into the memory.

50. Explain how priority of interrupt will be implemented? [WBSCTE 2019]
Answer:
A device's interrupt priority is selected based on two criteria: its maximum interrupt *latency* requirements and the device driver's interrupt *execution time*. The interrupt latency requirement is the maximum time within which an interrupt must be serviced. (If it is not serviced in this time, some event is lost or performance is seriously degraded.) The interrupt execution time is the number of machine cycles required by the device driver to service the interrupt. Interrupts with a short interrupt latency time must have a short interrupt service time.

51. What is non-volatile memory? [WBSCTE 2022]
Answer:
Non-volatile memory (NVM) or non-volatile storage is a type of computer memory that can retain stored information even after power is removed.

52. What is logical address? [WBSCTE 2022]
Answer: A logical address is an address that is generated by the CPU during program execution. The logical address is a virtual address as it does not exist physically, and therefore, it is also known as a ***Virtual Address***.

53. Which memory stores instruction which is required to start a computer? [WBSCTE 2022]

Answer:
ROM is a type of memory circuitry that holds the computer's startup routine.

54. What is the RAID system? [WBSCTE 2022]

Answer:
RAID (redundant array of independent disks) is a way of storing the same data in different places on multiple hard disks or solid-state drives (SSDs) to protect data in the case of a drive failure.

55. What is Cache memory? [WBSCTE 2022]

Answer:
Cache memory is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data.

56. What do you mean by the write-back policy? [WBSCTE 2022]

Answer:
Write back is a storage method in which data is written into the cache every time a change occurs, but is written into the corresponding location in main memory only at specified intervals or under certain conditions.

Long Answer Type Questions

1. a) What is cache memory. [WBSCTE 2005, 2006, 2007, 2008, 2009, 2010, 2010, 2017]

b) What are its advantages and uses? [WBSCTE 2005, 2006, 2007, 2009, 2010]

c) How information is stored and received from cache memory? [WBSCTE 2005]

OR,

Discuss different types of mapping functions used to represent cache memory with suitable diagram. [WBSCTE 2010, 2016, 2021]

OR,

Describe mapping techniques of cache memory. [WBSCTE 2015, 2017, 2019]

d) If in a byte addressable system the main memory size is 32 GB, associative cache size is 32 KB and main memory block size is 1 KB, find the number of tag bits in the physical address. [WBSCTE 2005, 2017]

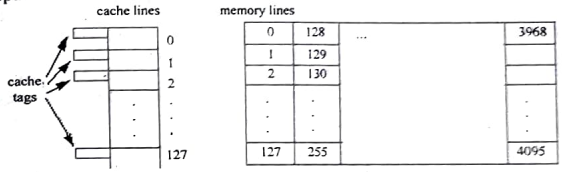
Answer:

a) *Refer to Article No. 5.2 of Unit at a glance.*

b) Cache (pronounced cash) memory is extremely fast memory that is built into a computer's central processing unit. The advantage of cache memory is that the CPU does not have to use the motherboard's system bus for data transfer. Whenever data must be

passed through the system bus, the data transfer speed slows to the motherboard's capability. The CPU can process data much faster by avoiding the bottleneck created by the system bus.

c) As a working example, suppose the cache has 27 = 128 lines, each with 24 = 16 words. Suppose the memory has a 16-bit address, so that 216 = 64K words are in the memory's address space.



Direct Mapping

Under this mapping scheme, each memory line j maps to cache line j mod 128 so the memory address looks like this:

	Tag	Line	Word
bits:	5	7	4

Here, the "Word" field selects one from among the 16 addressable words in a line. The "Line" field defines the cache line where this memory line should reside. The "Tag" field of the address is then compared with that cache line's 5-bit tag to determine whether there is a hit or a miss. If there's a miss, we need to swap out the memory line that occupies that position in the cache and replace it with the desired memory line.

E.g., Suppose we want to read or write a word at the address 357A, whose 16 bits are 0011010101111010. This translates to Tag = 6, line = 87, and Word = 10 (all in decimal). If line 87 in the cache has the same tag (6), then memory address 357A is in the cache. Otherwise, a miss has occurred and the contents of cache line 87 must be replaced by the memory line 001101010111 = 855 before the read or write is executed.

Direct mapping is the most efficient cache mapping scheme, but it is also the least effective in its utilization of the cache - that is, it may leave some cache lines unused.

Associative Mapping

This mapping scheme attempts to improve cache utilization, but at the expense of speed. Here, the cache line tags are 12 bits, rather than 5, and any memory line can be stored in any cache line. The memory address looks like this:

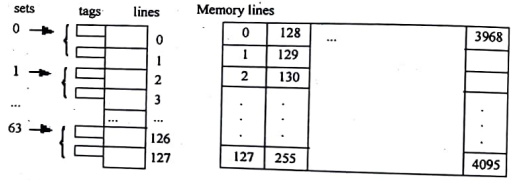
	Tag	Word
bits:	12	4

Here, the "Tag" field identifies one of the 2 12 = 4096 memory lines; all the cache tags are searched to find out whether or not the Tag field matches one of the cache tags. If so, we have a hit, and if not there's a miss and we need to replace one of the cache lines by

this line before reading or writing into the cache. (The "Word" field again selects one from among 16 addressable words (bytes) within the line.)

For example, suppose again that we want to read or write a word at the address 357A, whose 16 bits are 0011010101111010. Under associative mapping, this translates to Tag = 855 and Word = 10 (in decimal). So we search all of the 128 cache tags to see if any one of them will match with 855. If not, there's a miss and we need to replace one of the cache lines with line 855 from memory before completing the read or write. The search of all 128 tags in the cache is time-consuming. However, the cache is fully utilized since none of its lines will be unused prior to a miss (recall that direct mapping may detect a miss even though the cache is not completely full of active lines).

Set-associative Mapping: This scheme is a compromise between the direct and associative schemes described above. Here, the cache is divided into sets of tags, and the set number is directly mapped from the memory address (e.g., memory line j is mapped to cache set j mod 64), as suggested by the diagram below:



The memory address is now partitioned to like this:

	Tag	Set	Word
bits:	6	6	4

Here, the "Tag" field identifies one of the 26 = 64 different memory lines in each of the 26 = 64 different "Set" values. Since each cache set has room for only two lines at a time, the search for a match is limited to those two lines (rather than the entire cache). If there's a match, we have a hit and the read or write can proceed immediately. Otherwise, there's a miss and we need to replace one of the two cache lines by this line before reading or writing into the cache. In set-associative mapping, when the number of lines per set is n, the mapping is called n-way associative. For instance, the above example is 2-way associative.

d) No. of Blocks = Cache size / Block-size
= 32 KB / 1 KB = 2^5

No. of Sets = $2^5 / 4 = 2^3$

Tag + Set offset + Byte offset = 32

≈ Tag + 3 + 5 = 32

≈ Tag = 32 − 8 = 24 **(Ans.)**

➲ **2. (a) What are the different types of memory?** **[WBSCTE 2007]**

(b) What are the roles of main memory? **[WBSCTE 2007, 2008]**

Answer:

a) Memory can be broadly classified into three main parts: the cache memory, the main memory and the auxiliary memory.

Refer to Article No. 5.1 of Unit at a glance.

b) The main memory is the central storage unit in a computer system. The computer can manipulate only data that is in main memory. Therefore, every program you execute and every file you access must be copied from a storage device into main memory. The amount of main memory on a computer is crucial because it determines how many programs can be executed at one time and how much data can be readily available to a program.

➲ **3. (a) What are the differences between SRAM and DRAM?** **[WBSCTE 2008, 2010, 2011, 2012]**

(b) What is meant by refreshing of DRAM? **[WBSCTE 2008, 2010]**

(c) How EEPROM operates? **[WBSCTE 2008, 2010]**

Answer:

a)

	Static RAM		Dynamic RAM
1.	Such kind of RAM retains the stored information as long as the power supply is on.	1.	Loses its stored information in a very short time (a few milliseconds) even though the power supply is on.
2.	Circuit consists of multiple transistors (generally 6) and two cross-connected inverters (latch).	2.	Circuit consists of lesser number of (generally 1) transistors and a capacitor.
3.	Retaining of information depends on the power supply. It holds information in flip-flops.	3.	Depends on how long can the capacitor retain its charge. Holds information as charge on a capacitor.
4.	Costlier due to the requirement of multiple transistors.	4.	Comparatively Cheaper.
5.	Less packing density	5.	High packing density.
6.	Faster	6.	Moderate speeds but slower than static RAMs.
7.	Power consumption is high.	7.	Consume less power.
8.	Don't need refreshing of the circuitry.	8.	Needs refreshing of the circuitry.
9.	Storage capacity in a single memory chip is less.	9.	Storage capacity in a single memory chip is more.
10.	Has a shorter read and write cycle.	10.	Has a larger read and write cycle.
11.	More user-friendly.	11.	Less user-friendly.

b) If the sense amplifier senses that the capacitor's charge is above the threshold value, it enables the bit line to a full voltage which in turn recharges the capacitor to the full charge that corresponds to logic value 1 i.e. recharging the capacitor to the full charge means that again '1' is restored in the cell.
If the capacitor's charge is below threshold value, it means that capacitor has no charge and thus logic value '0' is stored in the cell.
So, in case of a dynamic RAM, reading the contents of the cell automatically refreshes its contents. All cells in a selected row are read at the same time, which refreshes the contents of the entire row.

c) *EEPROM (Electrically Erasable PROM):*
EEPROM chips do not have to be removed from the circuit board for erasure. The contents of EEPROM can be both erased & programmed electrically. It is possible to erase the cell contents selectively on a byte-by-byte basis. In EEPROM, it is possible to read & write the contents of a single cell. EPROM memory cells use floating gate technology.

➲ **4. (a) What are the differences between main memory and secondary memory?**
(b) What are the differences between ROM and RAM—explain with example.
[WBSCTE 2009]

Answer:

a)

Main Memory	Secondary Memory
1. Main memory is able to communicate directly with the CPU.	1. Secondary memory is not able to communicate directly with the CPU. It is always accessed through main memory.
2. Main memory is much faster than secondary memory.	2. Secondary memory is slower than main memory.
3. Main memory is costlier than secondary memory.	3. Secondary memory is cheaper than main memory.

b) Random Access Memory is a form of memory that can be read without having to begin at the first address, then the second address, then the third, and so on. This is a carryover from when most memory was Serial Access, such as magnetic tape, paper tape, or punched cards. The on-board memory computers use for temporary storage is Random Access Memory, but on chips, instead of a hard disc drive.
Read Only Memory is exactly what the name implies, it can only be read, not written to. A CD-ROM is a form of memory, as is a chip on the motherboard which is used to store instructions for the Central Processor Unit. RAM is your computer's temporary storage space. RAM is really the computer's short-term memory.

➲ **5. (a) What are the differences between ROM and PROM?**
(b) How EEPROM operates? [WBSCTE 2010]

Answer:

a)

ROM	PROM
i) Non-programmable.	i) Programmable.
ii) not flexible (because data cannot be changed).	ii) flexible (Field-programmable i.e. can be programmed in any place of work. On the other hand, if the programming can be done only at the factories or by masking, then it is known as factory-programmable).
iii) slower.	iii) faster.
iv) economical only when produced in large volumes.	iv) economical even when produced in small volumes.
v) used in PC's.	v) used mainly for research & development purposes.
vi) writing of cell contents mainly done by 'masking' mechanisms.	vi) mainly done electrically.

b) *EEPROM (Electrically Erasable PROM):*
EEPROM chips do not have to be removed from the circuit board for erasure. The contents of EEPROM can be both erased & programmed electrically. It is possible to erase the cell contents selectively on a byte-by-byte basis. In EEPROM, it is possible to read & write the contents of a single cell. EPROM memory cells use floating gate technology.

➲ **6. Discuss memory hierarchy with suitable block diagram and also discuss the cost, speed and size considering the memory hierarchy.** [WBSCTE 2010]

OR,

What is meant by memory organization? [WBSCTE 2010]

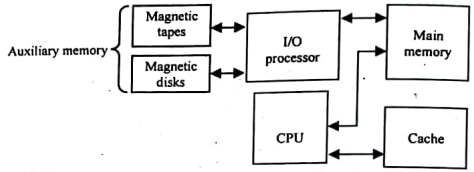
OR,

Discuss the memory hierarchy model with suitable diagram. [WBSCTE 2011]

OR,

What do you mean by memory hierarchy? [WBSCTE 2013]

Answer:



The total memory capacity of a computer can be visualized as being a hierarchy of components.

(a) The ***magnetic tape*** being the slowest component in the hierarchy lies at the bottom. These tapes are used to store removable files.

(b) Next lies the ***magnetic disks*** used as backup storage.

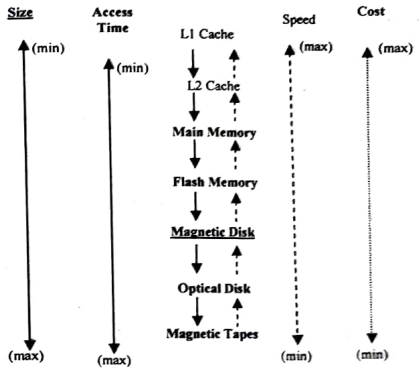
(c) In the central position of the hierarchy, lies the ***main memory***. Main memory generally stores the programs that are currently needed by the CPU.

(d) In between the main memory and the CPU lies the ***cache memory***. Cache memory is the fastest, smallest but the most expensive among the all memory devices. It stores the program segments and data that are needed frequently by CPU in current executions.

(e) The ***I/O processor*** manages data transfer between auxiliary and main memory.

Basic Objectives of Memory Hierarchy: The basic objective of using a memory hierarchy is to obtain the highest-possible access and transfer speed, maximum storage capacity, while minimizing the total cost of the entire memory system.

Factors on which Memory Hierarchy depends: There are various factors on which the basic hierarchy of memory depends. These are cost, storage capacity (size), and speed and access time.



➲ **7. Differentiate virtual memory and cache memory.**

[WBSCTE 2011, 2013, 2014, 2015, 2016, 2021]

Answer:

Comparison between Virtual Memory and Cache Memory:

	Virtual Memory	**Cache Memory**
• ***Definition***	Virtual memory is an abstraction of the main memory. It extends the available memory of the computer by storing the inactive parts of the content RAM on a disk. It fetches it back to the RAM when the content is required.	Cache memory is used to store frequently accessed data in order to quickly access the data whenever it is required. They both are conceptually the same; however they mainly differ in the matter of implementation.
• ***Purpose***	It extends the memory capacity of a computer beyond the one that is installed.	It reduces the amount of time needed to access the data.
• ***Speed***	It operates in the millisecond range.	It operates in the nanosecond range.
• ***Control mechanism***	Managed by the operating system	Managed automatically by the hardware
• ***Component***	It is a part of the hard drive (secondary storage).	Located on the processor itself

8. What is locality of references? Explain the concept of cache memory with it. A block set-associative cache consists of 64 blocks divided into 4 block sets. The main memory contains 4096 blocks, each consisting of 128 words of 16-bit length:
(a) How many bits are there in main memory?
(b) How many bits are there in each of the TAG, SET and word fields of the memory address generated by CPU? [WBSCTE 2012]

Answer:
Locality of reference lets a large program store its most essential and frequently used parts for quick access, instead of making your computer constantly access that code from your hard disk.
A word processor is a good example. Feature-rich word processors like Microsoft Word are several megabytes in size, but the part that waits for you to type something so it can display it on your computer's monitor can easily fit in cache memory.

a) The cache is divided into 16 sets of 4 lines each. Therefore, 4 bits are needed to identify the set number. Main memory consists of 4K = 212 blocks. Therefore, the set plus tag lengths must be 12 bits and therefore the tag length is 8 bits. Each block contains 128 words. Therefore, 7 bits are needed to specify the word.

b) TAG, SET and word fields of the memory address =

TAG	SET	WORD
8	4	7

9. How hit ratio depends on different types of mapping? [WBSCTE 2015, 2019]
Answer:
Direct Mapped Cache: The direct mapped cache is the simplest form of cache and the easiest to check for a hit. Since there is only one possible place that any memory location can be cached, there is nothing to search; the line either contains the memory information we are looking for, or it doesn't.
Unfortunately, the direct mapped cache also has the worst performance, because again there is only one place that any address can be stored. Consider a 512 KB level 2 cache and 64 MB of system memory. The cache has 16,384 lines (assuming 32-byte cache lines) and so each one is shared by 4,096 memory addresses. In the absolute worst case, imagine that the processor needs 2 different addresses (call them X and Y) that both map to the same cache line, in alternating sequence (X, Y, X, Y). This could happen in a small loop if we were unlucky. The processor will load X from memory and store it in cache. Then it will look in the cache for Y, but Y uses the same cache line as X, so it won't be there. So Y is loaded from memory, and stored in the cache for future use. But then the processor requests X, and looks in the cache only to find Y. This conflict repeats over and over. The net result is that the hit ratio here is 0%. This is a worst case scenario, but in general the performance is worst for this type of mapping.

Fully Associative Cache: The fully associative cache has the best hit ratio because any line in the cache can hold any address that needs to be cached. This means the problem seen in the direct mapped cache disappears, because there is no dedicated single line that an address must use.
However (you knew it was coming), this cache suffers from problems involving searching the cache. If a given address can be stored in any of 16,384 lines, how do you know where it is? Even with specialized hardware to do the searching, a performance penalty is incurred. And this penalty occurs for all accesses to memory, whether a cache hit occurs or not, because it is part of searching the cache to determine a hit. In addition, more logic must be added to determine which of the various lines to use when a new entry must be added (usually some form of a "least recently used" algorithm is employed to decide which cache line to use next). All this overhead adds cost, complexity and execution time.

N-Way Set Associative Cache: The set associative cache is a good compromise between the direct mapped and set associative caches. Let's consider the 4-way set associative cache. Here, each address can be cached in any of 4 places. This means that in the example described in the direct mapped cache description above, where we accessed alternately two addresses that map to the same cache line, they would now map to the same cache set instead. This set has 4 lines in it, so one could hold X and another could hold Y. This raises the hit ratio from 0% to near 100%! Again an extreme example, of course. As for searching, since the set only has 4 lines to examine this is not very complicated to deal with, although it does have to do this small search, and it also requires additional circuitry to decide which cache line to use when saving a fresh read from memory. Again, some form of LRU (least recently used) algorithm is typically used.

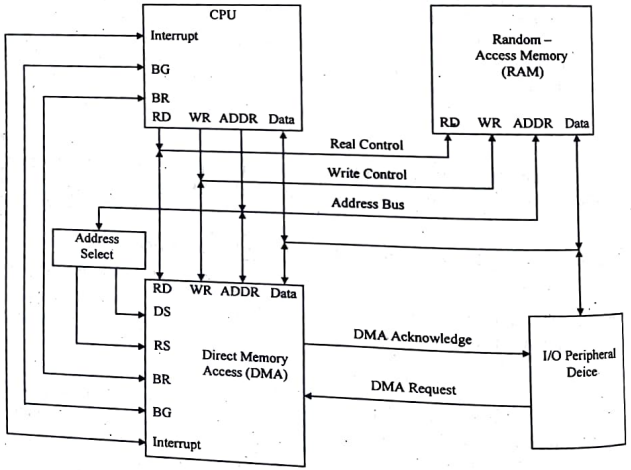
10. a) Differentiate programmed IO and Interrupt briefly. [WBSCTE 2021]
b) Describe DMA mode of data transfer in details with suitable diagram. [WBSCTE 2019, 2021]

Answer:
a) Difference between Programmed and Interrupt Initiated I/O:

Programmed I/O	Interrupt Initiated I/O
Data transfer is initiated by the means of instructions stored in the computer program. Whenever there is a request for I/O transfer the instructions are executed from the program.	The I/O transfer is initiated by the interrupt command issued to the CPU.
The CPU stays in the loop to know if the device is ready for transfer and has to continuously monitor the peripheral device.	There is no need for the CPU to stay in the loop as the interrupt command interrupts the CPU when the device is ready for data transfer.
This leads to the wastage of CPU cycles as CPU	The CPU cycles are not wasted as

Programmed I/O	Interrupt Initiated I/O
remains busy needlessly and thus the efficiency of system gets reduced.	CPU continues with other work during this time and hence this method is more efficient.
CPU cannot do any work until the transfer is complete as it has to stay in the loop to continuously monitor the peripheral device.	CPU can do any other work until it is interrupted by the command indicating the readiness of device for data transfer
Its module is treated as a slow module.	Its module is faster than programmed I/O module.
It is quite easy to program and understand.	It can be tricky and complicated to understand if one uses low level language.
The performance of the system is severely degraded.	The performance of the system is enhanced to some extent.

b)



DMA Transfer Sequence:

- Peripheral device sends DMA request
- DMA controller activates BR
- CPU finishes current bus cycle and grant the bus by activating BG
- DMA puts current address to the address bus and activates RD or WR accordingly
- And acknowledge peripheral
- Then peripheral puts data to (or reads data from) the bus
- Thus peripheral directly read or write memory
- For each word transferred DMA increment address and decrement word count register
- If word count is not zero DMA checks request line coming from peripheral
  - If active (fast devices) initiate the second transfer immediately
  - Otherwise disable BR
- If word count is zero
  - DMA stop transfer, disable BR and inform CPU the termination of data transfer
- Zero value in word count indicates successful data transfer
- DMA can have even more than one channel
- DMA commonly used in devices like magnetic disks and screen display.

11. a) Discuss the various mapping techniques used in cache memory. [WBSCTE 2021, 2022]

b) What is virtual memory? How does it work? [WBSCTE 2022]

c) How can you interface RAM and the ROM EPROM to microprocessor 8086? What is the use of EPROM? [WBSCTE 2022]

Answer:

a) CACHE MAPPING TECHNIQUES: Cache mapping is the method by which the contents of main memory are brought into the cache and referenced by the CPU. The mapping method used directly affects the performance of the entire computer system.

- **Direct mapping** – main memory locations can only be copied into one location in the cache. This accomplished by dividing main memory into pages that correspond in size with the cache (fig. a).
- **Fully associative mapping** – Fully associative cache mapping is the most complex, but it is most flexible with regards to where data can reside. A newly read block of main memory can be placed anywhere in a fully associative cache. If the cache is full, a replacement algorithm is used to determine which block in the cache gets replaced by the new data (fig. b).
- **Set associative mapping** – set associative cache mapping combines the best of direct and associative cache mapping techniques. As with a direct mapped cache, blocks of main memory data will still map into as specific set, but they can now be in any N-cache block frames within each set (fig. c).

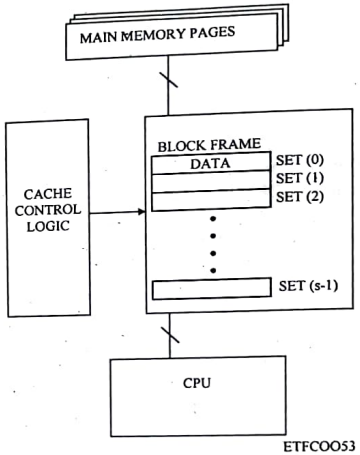


Fig: (a) Example of direct mapping used in cache memory

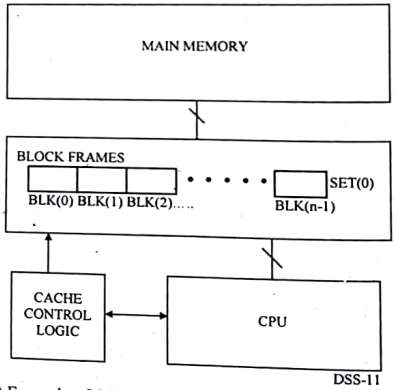


Fig: (b) Example of fully associated mapping in cache memory

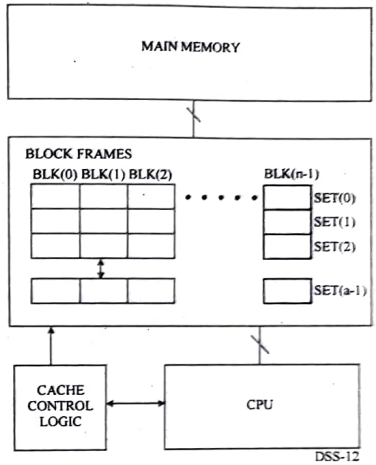


Fig: (c) Example of set association mapping used in cache

b) 1st Part:

Virtual memory is a memory management technique where secondary memory can be used as if it were a part of the main memory. Virtual memory is a common technique used in a computer's operating system (OS).

Virtual memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily transferring data from random access memory (**RAM**) to disk storage. Mapping chunks of memory to disk files enables a computer to treat secondary memory as though it were main memory.

2nd Part:

Virtual memory uses both hardware and software to operate. When an application is in use, data from that program is stored in a physical address using RAM. A memory management unit (**MMU**) maps the address to RAM and automatically translates addresses. The MMU can, for example, map a logical address space to a corresponding physical address.

If, at any point, the RAM space is needed for something more urgent, data can be swapped out of RAM and into virtual memory. The computer's memory manager is in

charge of keeping track of the shifts between physical and virtual memory. If that data is needed again, the computer's MMU will use a **context switch** to resume execution. While copying virtual memory into physical memory, the OS divides memory with a fixed number of addresses into either pagefiles or **swap files**. Each page is stored on a disk, and when the page is needed, the OS copies it from the disk to main memory and translates the virtual addresses into real addresses.

However, the process of swapping virtual memory to physical is rather slow. This means using virtual memory generally causes a noticeable reduction in performance. Because of swapping, computers with more RAM are considered to have better performance.

c) 1st Part:

The general procedure of static memory interfacing with 8086 is briefly described as follows:

1. Arrange the available memory chips so as to obtain 16-bit data bus width. The upper 8-bit bank is called 'odd address memory bank' and the lower 8-bit bank is called 'even address memory bank'.
2. Connect available memory address lines of memory chips with those of the microprocessor and also connect the memory RD and WR inputs to the corresponding processor control signals. Connect the 16-bit data bus of the memory bank with that of the microprocessor 8086.
3. The remaining address lines of the microprocessor, BHE and A0 are used for decoding the required chip select signals for the odd and even memory banks. CS of memory is derived from the O/P of the decoding circuit. The address map of the system should be continuous as far as possible, i.e. there should be no windows in the map. A memory location should have a single address corresponding to it, i.e. absolute decoding should be preferred, and minimum hardware should be used for decoding. Mostly, linear decoding are used to minimize the required hardware.

2nd Part:

The Erasable Programmable Read-Only Memory (EPROM) is a data circuit that retains its memory even when power is off. Security system manufacturers use EPROMs to hold both the operating system and the program for the access control panel.

12. Write short notes on the following:

a) Virtual Memory [WBSCTE 2019]

b) Memory Interleaving [WBSCTE 2021]

c) Loop Buffer [WBSCTE 2021]

Answer:

a) ***Refer to Question No. 11.b) of Long Answer Type Questions.***

b) Main memory (random-access memory, RAM) is often made up of a collection of DRAM memory chips, with several chips grouped to form a memory bank. The memory banks can then be laid out to interleave using a memory controller that supports interleaving.

In turn, interleaved memory addresses are assigned to each memory bank. In a system with two interleaved memory banks (assuming word-addressable memory), for example, if logical address 32 belongs to bank 0, logical address 33 belongs to bank 1, logical address 34 belongs to bank 0, and so on. When there are n banks and a memory location (i sits in bank i mod n.), The memory is said to be n-way interleaved.

For example, if we have four memory banks (four-way interleaved memory), each with 256 bytes, the Block Oriented method (no interleaving) will allocate An instruction pipeline that can request both instructions and operands from the main memory at the same time, which is not possible with regular memory access.

Similarly, an arithmetic pipeline necessitates the simultaneous retrieval of two operands from the main memory. The first bank has virtual addresses ranging from 0 to 255, whereas the second bank has addresses ranging from 256 to 511. However, in Interleaved memory, virtual address 0 corresponds to the first memory bank, virtual address 1 corresponds to the second memory bank, virtual address 2 corresponds to the third memory bank, virtual address 3 corresponds to the fourth memory bank, and virtual address 4 corresponds to the first memory bank again.

As a result, the CPU can access different parts without waiting for memory to be cached. Several memory banks supply data in turn.

Memory interleaving is a memory speed-up technique. It is a procedure that improves the system's efficiency, speed, and dependability.

Types of Interleaved Memory:

1. **High Order Interleaving**: The memory address's most significant bits indicate which memory banks contain a particular spot in high-order memory interleaving. However, the memory banks are chosen by the memory address's least significant bits in low order interleaving.

2. **Low Order Interleaving**: One significant difference between high order and low order interleaving is that consecutive memory locations are found in the same memory module in high order interleaving. However, sequential memory locations are located in consecutive banks with low order interleaving.

c) *Refer to Question No. 47 of Short Answer Type Questions.*

Question Paper 2019 (December)

1. A. Choose the correct answer from the given alternatives (any ten):

i) Which of the following comment about the IR is true?–

(a) It is used to count the number of instructions

(b) It is a cell in ROM

(c) During execution of the current instruction, its content changes

✓(d) Opcode fetched from memory stored in IR

ii) The speed imbalance between memory access & CPU operation can be reduced by –

(a) Cache memory ✓ (b) memory interleaving

(c) reducing memory size (d) virtual memory

iii) The sequence of events that happen during a typical fetch operation is –

(a) PC → MDR → Memory → IR

✓ (b) PC → MAR → MEMORY → MDR → IR

(c) PC → MEMORY → IR

(d) MAR → MDR → IR

iv) Negative numbers can be represented in –

(a) Sign-magnitude form (b) 1's complemented form

(c) 2's complemented form ✓ (d) all of the above

v) The instruction ADD is _____ address instruction. –

(a) 0 (b) 1 ✓ (c) 2 (d) 3

vi) The exponent of a floating point number is represented in excess-N code (biased) so that –

(a) the dynamic range is large

(b) the precision is high

✓ (c) the smallest number is represented by all zeroes

(d) overflow is avoided

vii) If negative numbers are stored in 2's complemented form, the range of numbers that can be stored in 8 bits is –

(a) –128 to +128
✓ (b) –128 to +127
(c) –127 to +128
(d) –127 to +127

viii) The three main components of a digital computer system are –

(a) Memory, IO, DMA
(b) ALU, CPU, Memory
✓ (c) CPU, Memory, IO
(d) Control Unit, ALU, Register

ix) The cost of storing a bit is minimum in –

(a) Cache memory
(b) Register
(c) RAM
✓ (d) Hard Disk

x) Which of the following statement(s) is true? –

(a) ROM is a read/write memory
(b) PC points to the last instruction that was executed
✓ (c) Stack works in the principle of FIFO
✓ (d) RAM is a Read/Write memory

xi) Tera is 2 to the power of –

(a) 9 (b) 20 (c) 30 ✓ (d) 40

xii) In immediate addressing the operand is placed –

(a) in the CPU register
✓ (b) after OP code in the instruction
(c) in memory
(d) in stack

B. Answer the following questions (any five):

i) What is write through policy?

See Unit: Memory and Digital Interfacing, Question No. 47 of Short Answer Type Questions.

ii) What is supercomputer?

See Unit: Structure of Computers, Question No. 65 of Short Answer Type Questions.

iii) What is TLB?

See Unit: Memory and Digital Interfacing, Question No. 42 of Short Answer Type Questions.

iv) What do mean by vectored interrupt?

See Unit: Memory and Digital Interfacing, Question No. 49 of Short Answer Type Questions.

v) What is the function of IO processor?

See Unit: Memory and Digital Interfacing, Question No. 50 of Short Answer Type Questions.

vi) What do you mean by Loop Buffer?

See Unit: Memory and Digital Interfacing, Question No. 48 of Short Answer Type Questions.

vii) What do mean by Replacement Algorithm?

See Unit: Memory and Digital Interfacing, Question No. 43 of Short Answer Type Questions.

2. Describe the following addressing modes with suitable example.

(i) Direct addressing mode, (ii) Indirect addressing mode, (iii) Indexed addressing mode, (iv) Based addressing mode, (v) Implied addressing mode.

See Unit: Introduction to Microprocessor Architecture, Question No. 1 of Long Answer Type Questions.

3. a) Describe Booth's algorithm with suitable architectural diagram.

b) Explain how priority of interrupt will be implemented?

a) See Unit: Micro Programmed Control, Question No. 6 of Long Answer Type Questions.

b) See Unit: Memory and Digital Interfacing, Question No. 51 of Short Answer Type Questions.

4. a) Write down the flowchart of both restoring and non-restoring division of integer numbers.

b) Write down the features of RISC architecture.

a) See Unit: Micro Programmed Control, Question No. 10.a) of Long Answer Type Questions.

b) See Unit: Micro Programmed Control, Question No. 28 of Long Answer Type Questions.

5. a) Explain different types of mapping technique used in cached memory.

b) How hit ratio depends on different types of mapping.

a) See Unit: Memory and Digital Interfacing, Question No. 1 of Long Answer Type Questions.

b) See Unit: Memory and Digital Interfacing, Question No. 9 of Long Answer Type Questions.

6. a) Explain the representation of floating point number in computer. Describe the IEEE format also.

b) What is NaN in floating point number representation? Give example.

See Unit: Structure of Computers, Question No. 13 of Long Answer Type Questions.

7. a) Explain the difference between Hardwired Control and Micro-programmed Control.

b) Explain the organization of control memory in Micro-programmed control with diagram.

See Unit: Micro Programmed Control, Question No. 2 of Long Answer Type Questions.

8. a) Describe DMA mode of data transfer in details with suitable diagram.

b) RISC architecture is more suitable for pipeline implementation-explain.

a) See Unit: Memory and Digital Interfacing, Question No. 10.b) of Long Answer Type Questions.

b) See Unit: Micro Programmed Control, Question No. 29 of Long Answer Type Questions.

9. a) Discuss Flynn's classification in details.

b) Explain instruction pipeline with suitable example and diagram.

a) See Unit: Micro Programmed Control, Question No. 29 of Long Answer Type Questions.

b) See Unit: Micro Programmed Control, Question No. 31 of Long Answer Type Questions.

10. Write shorts notes on the following (any two):

a) Generations of computer

b) Space time diagram

c) Array processor

d) Virtual memory

a) See Unit: Structure of Computers, Question No. 16.f) of Long Answer Type Questions.

b) See Unit: Micro Programmed Control, Question No. 35 of Long Answer Type Questions.

c) See Unit: Micro Programmed Control, Question No. 43.a) of Long Answer Type Questions.

d) See Unit: Memory and Digital Interfacing, Question No. 12.a) of Long Answer Type Questions.

Question Paper 2021 (March)

1. A. Choose the correct answer from the given alternatives (Any Ten):

i) The instruction 1111 111100001100 is a

✓ (a) direct memory reference instruction (b) indirect memory reference instruction
(c) register reference instruction (d) input output instruction

ii) 01110000 represents

✓ (a) 0 (b) NaN (c) $+\infty$ (d) $-\infty$

iii) The largest floating point number that can be represented by 8 bit is

✓ (a) 01111111 (b) 11111111 (c) 01101111 (d) 01111110

iv) If n is number of bits in exponent, the bias number can be calculated as

✓ (a) 2^n-1 (b) 2^n (c) 2^n-1 (d) $2^{n-1}-1$

v) In Booth's algorithm, if the multiplier has n bits then the multiplicand should have

(a) 1 bit (b) n bits ✓ (c) n+1 bits (d) 2n bits

vi) k-way set associative means

✓ (a) k blocks are present in a set (b) k sets are present in a block
(c) k sets are present in the cache (d) none of these

vii) The three main components of a digital computer system are

(a) Memory, IO, DMA (b) ALU, CPU, Memory
(c) CU, ALU, Register ✓ (d) CPU, Memory, IO

viii) The second generation of computer used

✓ (a) transistors (b) IC (c) vacuum tube (d) LSI

ix) Processors of all computers, whether micro, mini or mainframe must have

(a) ALU (b) Primary storage (c) Control Unit ✓ (d) all of these

x) In Which addressing mode is operand specified in the instruction itself?

(a) Register mode (b) Immediate mode
(c) Direct Address mode (d) Index Addressing mode

Answer: (implied addressing mode)

xi) The instruction LOAD is a

(a) zero-address instruction ✓ (b) one-address instruction
(c) two-address instruction (d) three-address instruction

xii) The number of fetch operation to execute instruction in immediate mode is

✓ (a) 0 (b) 1 (c) 2 (d) none of these

B. Answer the following questions (any five):

i) What is Program Counter?

See Unit: Micro Programmed Control Question No. 36 of Short Answer Type Questions.

ii) What is IO processor?

See Unit: Structure of Computers Question No. 66 of Short Answer Type Questions.

iii) Write full form of RISC and CISC.

See Unit: Micro Programmed Control Question No. 37 of Short Answer Type Questions.

iv) What is a super computer?

See Unit: Structure of Computers Question No. 65 of Short Answer Type Questions.

v) How does a computer differ from a calculator?

See Unit: Structure of Computers Question No. 63 of Short Answer Type Questions.

vi) What is bus?

See Unit: Structure of Computers Question No. 64 of Short Answer Type Questions.

vii) What do you mean by Effective Address?

See Unit: Introduction to Microprocessor Architecture Question No. 12 of Short Answer Type Questions.

2. Write short notes on the followings (any two):

a) IO processor

b) Vector Processing

c) Memory Interleaving

d) Loop Buffer

a) See Unit: Structure of Computers Question No. 15.g) of Long Answer Type Questions.

b) See Unit: Structure of Computers Question No. 43.b) of Long Answer Type Questions.

c) See Unit: Memory and Digital Interfacing Question No. 11.b) of Long Answer Type Questions.

d) See Unit: Memory and Digital Interfacing Question No. 11.c) of Long Answer Type Questions.

3. a) Explain different types of mapping technique used in cache memory.

b) Differentiate virtual memory and cache memory.

a) See Unit: Memory and Digital Interfacing Question No. 1.c) of Long Answer Type Questions.

b) See Unit: Memory and Digital Interfacing Question No. 7 of Long Answer Type Questions.

4. What do you mean by pipeline hazards/conflicts? Discuss the different types of hazards being observed and also explain the possible solutions.

See Unit: Micro Programmed Control Question No. 33 of Long Answer Type Questions.

5. a) What are the major characteristics of RISC architecture?

b) What do you mean by speedup ratio? What are the reasons for which theoretical maximum speedup cannot be obtained in reality?

See Unit: Micro Programmed Control Question No. 36 of Long Answer Type Questions.

6. a) Differentiate programmed IO and Interrupt briefly.

b) Describe DMA mode of data transfer in details with suitable diagram.

See Unit: Micro Programmed Control Question No. 10 of Long Answer Type Questions.

7. Describe the following addressing modes with suitable example.

i) Register addressing mode

ii) Indirect addressing mode

iii) Indexed addressing mode

iv) Base addressing mode

v) Immediate addressing mode

See Unit: Introduction to Microprocessor Architecture Question No. 1 of Long Answer Type Questions.

8. a) Describe Booth's algorithm with suitable block diagram and flowchart.

b) Show the steps of multiplication performed by using Booth's algorithm of 7X –5.

See Unit: Micro Programmed Control Question No. 17 of Long Answer Type Questions.

9. a) Explain the difference between Hardwired Control and Microprogrammed control.

b) What do you mean by horizontal and vertical microprogramming? Compare these two ways of microprogramming.

See Unit: Micro Programmed Control Question No. 3 of Long Answer Type Questions.

Question Paper 2022 (June)

1. Choose the correct alternatives (any ten):

i) The number successful accesses to memory stated as a fraction is called as_____.

(a) Access rate (b) Success rate ✓ (c) Hit rate (d) Miss rate

ii) The final addition sum of the numbers, 0.110 & 0110 is _____.

✓ (a) 1101 (b) 1111 (c) 1001 (d) 1010

iii) What does CSA stands for?

(a) Computer Service Architecture ✓ (b) Computer Speed Addition

(c) Carry Save Addition (d) None of these

iv) Individual control word of the micro routine are called as

(a) Micro task ✓ (b) Micro instruction

(c) Micro operation (d) Micro Command

v) Which of the following circuit convert the binary data into a decimal?

(a) Decoder (b) Encoder ✓ (c) Code converter (d) Multiplexer

vi) The situation wherein the data of operands are not available is called_____

✓ (a) Data hazard (b) Stock

(c) Deadlock (d) Structural hazard

vii) What is the full form of CISC?

(a) Complex Instruction Sequential Compilation

(b) Complete Instruction Sequential Compilation

(c) Computer Integrated Sequential Compiler

✓ (d) Complex Instruction Set Computer

viii) The alternate way of writing the instruction, ADD #5, R1 is

(a) ADD [5], [R1] ✓ (b) ADDI 5, R1

(c) ADDIME 5, [R1] (d) There is no other way

ix) In order to read multiple bytes of a row at the same time , we make use of

✓ (a) Memory extension (b) Cache

(c) Shift register (d) Latch

x) In full adders the sum circuit is implemented using_____.

(a) And & or gates (b) NAND gate ✓ (c) XOR (d) XNOR

xi) Computer address bus is

(a) Unidirectional ✓ (b) Bidirectional

(c) Multidirectional (d) None of the above

xii) Which of the following computer bus connects the CPU to a memory on the system board?

(a) Expansion bus (b) Width bus

✓ (c) System bus (d) None of the above

xiii) The instructions that are used for reading an input port and writing an output port respectively are

(a) MOV, XCHG (b) MOV, IN (c) IN, MOV ✓ (d) IN, OUT

xiv) Micro operation is shown as

✓ (a) R1←R2 (b) R1+R2 (c) Both (d) None

xv) An interrupt that can be temporarily ignored is

(a) Vectored interrupt (b) Non – maskable interrupt

✓ (c) Maskable interrupt (d) High priority interrupt

2. Fill in the blanks (Any ten):

i) ***Secondary memory*** is used to store data, instructions are results permanently for future use.

ii) ***Virtual memory*** is generally used to increase the apparent size of physical memory .

iii) Gray Code is also called as ***reflected binary code***.

iv) Instruction register stores ***the instruction currently being executed***.

v) A high speed memory is placed between the CPU and the primary memory is known as ***cache memory***.

vi) I/O address in 8086 is ***16*** bit.

vii) Technique that automatically move programs and data blocks into the physical memory when they are required for execution are called ***virtual memory technique***.

viii) Hit ratio is maximum in ***set associative*** mapping .

ix) The bias value for single-precision floating point numbers is ***127***

x) MOV AX. [2A50] is an example of ***direct*** addressing mode.

xi) Loop unrolling is a technique to improve ***performance***.

xii) Page table resides in ***main memory***.

xiii) Microinstruction consists of ***micro-operations***.

xiv) The smallest entity of memory is called ***cell***.

xv) A source program is usually in ***high-level*** language.

3. Answer the following question (any ten):

i) How control unit controls other units?

See Unit: Micro Programmed Control Question No. 39 of Short Answer Type Questions.

ii) Give an example of a 4 bit, 8 bit, 16 bit and 32 bit microprocessor.

See Unit: Introduction to Microprocessor Architecture Question No. 13 of Short Answer Type Questions.

iii) What is Bus?

See Unit: Structure of Computers Question No. 64 of Short Answer Type Questions.

iv) What is MAR and MDR?

See Unit: Structure of Computers Question No. 67 of Short Answer Type Questions.

v) What is register?

Answer:

See Unit: Structure of Computers Question No. 68 of Short Answer Type Questions.

vi) What is interrupt?

See Unit: Introduction to Microprocessor Architecture Question No. 14 of Short Answer Type Questions.

vii) What is non-volatile memory?

See Unit: Memory and Digital Interfacing Question No. 52 of Short Answer Type Questions.

viii) What is logical address?

See Unit: Memory and Digital Interfacing Question No. 53 of Short Answer Type Questions.

ix) Which is an error-detecting code?

See Unit: Structure of Computers Question No. 69 of Short Answer Type Questions.

x) What is the logic shift?

See Unit: Structure of Computers Question No. 70 of Short Answer Type Questions.

xi) What type of device converts digital signal into a form that is intelligible to the user?

See Unit: Structure of Computers Question No. 71 of Short Answer Type Questions.

xii) Which memory stores instruction which is required to start a computer?

See Unit: Memory and Digital Interfacing Question No. 54 of Short Answer Type Questions.

xiii) Define clock rate?

See Unit: Structure of Computers Question No. 72 of Short Answer Type Questions.

xiv) What is the RAID system?

See Unit: Memory and Digital Interfacing Question No. 55 of Short Answer Type Questions.

4. Answer the question (any six) 2×6 = 12

i) What are the three main elements of the control unit?

See Unit: Micro Programmed Control Question No. 40 of Short Answer Type Questions.

ii) What is Cache memory?

See Unit: Memory and Digital Interfacing Question No. 56 of Short Answer Type Questions.

iii) What is control memory address?

See Unit: Micro Programmed Control Question No. 41 of Short Answer Type Questions.

iv) What is the 2's complement representation of –6?

See Unit: Micro Programmed Control Question No. 42 of Short Answer Type Questions.

v) What is clock signal in COA?

See Unit: Structure of Computers Question No. 73 of Short Answer Type Questions.

vi) Is USB is a bus?

See Unit: Structure of Computers Question No. 74 of Short Answer Type Questions.

vii) Draw the block diagram of the half adder.

See Unit: Micro Programmed Control Question No. 43 of Short Answer Type Questions.

viii) Draw a multiplication circuit diagram.

See Unit: Micro Programmed Control Question No. 44 of Short Answer Type Questions.

ix) What's the difference between interrupt service routine and subroutine?

See Unit: Introduction to Microprocessor Architecture Question No. 15 of Short Answer Type Questions.

x) What do you mean by the write-back policy?

See Unit: Memory and Digital Interfacing Question No. 57 of Short Answer Type Questions.

xi) What is RISC Pipeline?

See Unit: Micro Programmed Control Question No. 45 of Short Answer Type Questions.

xii) What size of MUXs are needed?

See Unit: Micro Programmed Control Question No. 46 of Short Answer Type Questions.

5. Answer the question (any one)

a) Explain the components of the Computer system and what is micro operation?

b) Represent $(12.625)_{10}$ in 32 bit floating point representation and what is odd parity checker?

c) Describe the Von- Neumann Architecture with diagram? Explain the Bus Structure with examples.

a) See Unit: Structure of Computers Question No. 14.a) of Long Answer Type Questions.

b) See Unit: Structure of Computers Question No. 21 of Long Answer Type Questions.

c) See Unit: Structure of Computers Question No. 14.b) of Long Answer Type Questions.

6. Answer the question (any one)

a) Describe the Flag Register of 8086 microprocessor.

b) Perform multiplication between 23 and 17 using fixed point multiplication algorithm.

c) What are the key characteristics of micro-programmed control? Explain different types of micro operation.

a) See Unit: Introduction to Microprocessor Architecture Question No. 2 of Long Answer Type Questions.

b) See Unit: Micro Programmed Control Question No. 22.a) of Long Answer Type Questions.

c) See Unit: Micro Programmed Control Question No. 22.b) of Long Answer Type Questions.

7. Answer the question (any one):

a) Discuss the various mapping techniques used in cache memory.

b) What is virtual memory? How does it work?

c) How can you interface RAM and the ROM EPROM to microprocessor 8086? What is the use of EPROM?

See Unit: Memory and Digital Interfacing Question No. 11 of Long Answer Type Questions.