SUBJECT: COMPUTER ORGANIZATION & ARCHITECTURE

STEP MATERIAL

UNIT I

SHORT ANSWERS

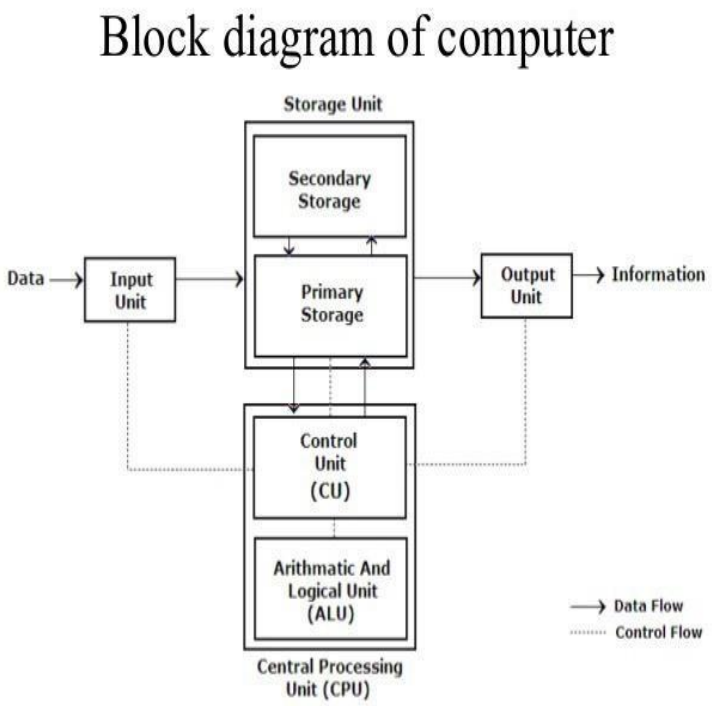
1)Define computer architecture?

Computer Architecture refers to those attributes of a system visible to a programmer or, put another way, those attributes that have direct impact on the logical execution of a program.For example, it is an **architectural** design issue whether a computer will have a multiply instruction.

2)Define computer organization?

Computer Organization refers to the operational units and their interconnections that realize the architectural specifications. **Organizational** attributes include those hardware details transparent to the programmer, such as control signals, interfaces between the computer and peripherals and memory technology used.

3)Draw the block diagram of digital computers?



4)Differences between computer architecture and computer organization?

Computer Architecture	Computer Organization
Computer Architecture is concerned with the way hardware components are connected together to form a computer system.	Computer Organization is concerned with the structure and 1ehavior of a computer system as seen by the user.

It acts as the interface between hardware and software.	It deals with the components of a connection in a system.
Computer Architecture helps us to understand the functionalities of a system.	Computer Organization tells us how exactly all the units in the system are arranged and interconnected.
A programmer can view architecture in terms of instructions, addressing modes and registers.	Whereas Organization expresses the realization of architecture.
While designing a computer system architecture is considered first.	An organization is done on the basis of architecture.
Computer Architecture deals with high-level design issues.	Computer Organization deals with low-level design issues.
Architecture involves Logic (Instruction sets, Addressing modes, Data types, Cache optimization)	Organization involves Physical Components (Circuit design, Adders, Signals, Periph

5)What is register transfer?

Data Transfer from one register to another register is represented in symbolic form by means of replacement operator. For instance, the following statement denotes a transfer of the data of register R1 into register R2.

$$R2 \leftarrow R1$$

6)what is instruction cycle?

A program residing in the memory unit of a computer consists of a sequence of instructions. These instructions are executed by the processor by going through a cycle for each instruction.

7)what are the Basic computer instruction cycle ?

In a basic computer, each instruction cycle consists of the following phases:

1. Fetch instruction from memory.
2. Decode the instruction.
3. Read the effective address from memory.
4. Execute the instruction.

8)How is arithmetic micro operation classified?

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

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

9) Define bus?

A digital system composed of many registers, and paths must be provided to transfer information from one register to another. The number of wires connecting all of the registers will be excessive if separate lines are used between each register and all other registers in the system.

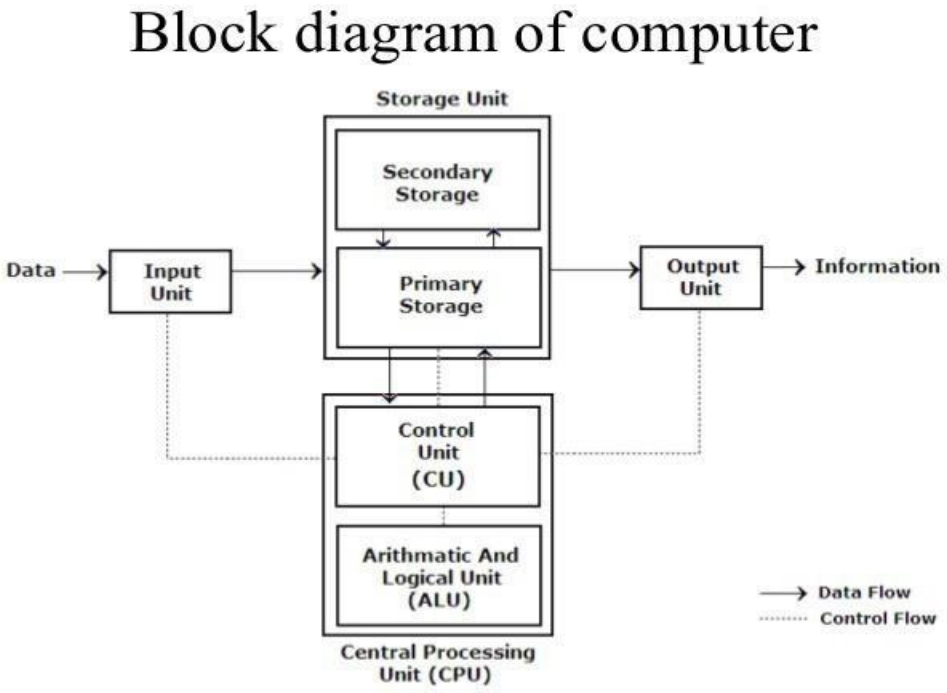
10)Define memory transfer?

- The transfer of information from a memory unit to the user end is called a **Read** operation.
- The transfer of new information to be stored in the memory is called a **Write** operation.
- A memory word is designated by the letter **M**.
- We must specify the address of memory word while writing the memory transfer operations.
- The **address register** is designated by **AR** and the **data register** by **DR**.

LONG ANSWERS:

1)Draw the basic block function of computer and explain in detail?

A digital computer is considered to be a calculating device that can perform arithmetic operations at enormous speed. It is defined as a device that operates upon information/data. To be able to process data the computer is made of various functional units to perform its specified task.



Input Unit:
Computers need to receive data and instruction in order to solve any problem. Therefore, we need to input the data and instructions into the computers. The input unit consists of one or more input devices. Keyboard is the one of the most

commonly used input device. Other commonly used input devices are the Mouse, Scanner, Microphone etc. All the input devices perform the following functions.
• Accept the data and instructions from the outside world.
• Convert it to a form that the computer can understand.
• Supply the converted data to the computer system for further processing.

Storage Unit:
The storage unit of the computer holds data and instructions that are entered through the input unit, before they are processed. It preserves the intermediate and final results before these are sent to the output devices.
a) **Primary Storage:** Stores and provides very fast. This memory is generally used to hold the program being currently executed in the computer, the data being received from the
input unit, the intermediate and final results of the program. The primary memory is temporary in nature.
b) **Secondary Storage:** Secondary storage is used like an archive. It stores several programs, documents, data bases etc. The programs that you run on the computer are first transferred to the primary memory before it is actually run. Whenever the results are saved, again they get stored in the secondary memory. The secondary memory is slower and cheaper than the primary memory. Some of the commonly used secondary memory devices are Hard disk, CD, etc.,

Memory Size:
All digital computers use the binary system, i.e. 0's and 1's. Each character or a number is represented by an 8-bit code. The set of 8 bits is called a byte. A character occupies 1-byte space. A numeric occupies 2-byte space. Byte is the space occupied in the memory.

Output Unit:
The output unit of a computer provides the information and results of a computation to outside world. Printers, Visual Display Unit (VDU) are the commonly used output devices. Other commonly used output devices are Speaker, Headphone, Projector etc.
Arithmetic Logical Unit:
All calculations are performed in the Arithmetic Logic Unit (ALU) of the computer. It also does comparison and takes decision. The ALU can perform basic operations such as addition, subtraction, multiplication, division, etc and does logic operations viz, >, <, =, 'etc.
Control Unit:
It controls all other units in the computer. The control unit instructs the input unit, where to store the data after receiving it from the user. It controls the flow of data and instructions from the storage unit to ALU.
Central Processing Unit:
The Control Unit (CU) and Arithmetic Logic Unit (ALU) of the computer are together known as the Central Processing Unit (CPU). The CPU is like brain performs the following functions:
• It performs all calculations.
• It takes all decisions.
•It controls all units of the computer.

2) Explain about register transfer language?

A digital computer system exhibits an interconnection of digital modules such as registers, decoders, arithmetic elements, and Control logic.

These digital modules are interconnected with some common data and control paths to form a complete digital system.

Moreover, digital modules are best defined by the registers and the operations that are performed on the data stored in them.

The operations performed on the data stored in registers are called **Micro-operations**.

The internal hardware organization of a digital system is best defined by specifying:

- The set of registers and the flow of data between them.
- The sequence of micro-operations performed on the data which are stored in the registers.
- The control paths that initiates the sequence of micro-operation

The **Register Transfer Language** is the symbolic representation of notations used to specify the sequence of micro-operations.

In a computer system, data transfer takes place between processor registers and memory and between processor registers and input-output systems. These data transfer can be represented by standard notations given below:

- Notations R0, R1, R2..., and so on represent processor registers.
- The addresses of memory locations are represented by names such as LOC, PLACE, MEM, etc.
- Input-output registers are represented by names such as DATA IN, DATA OUT and so on.
- The content of register or memory location is denoted by placing square brackets around the name of the register or memory location.

3)Describe about bus system for 4 register and three state gate bus?

A digital system composed of many registers, and paths must be provided to transfer information from one register to another. The number of wires connecting all of the registers will be excessive if separate lines are used between each register and all other registers in the system.

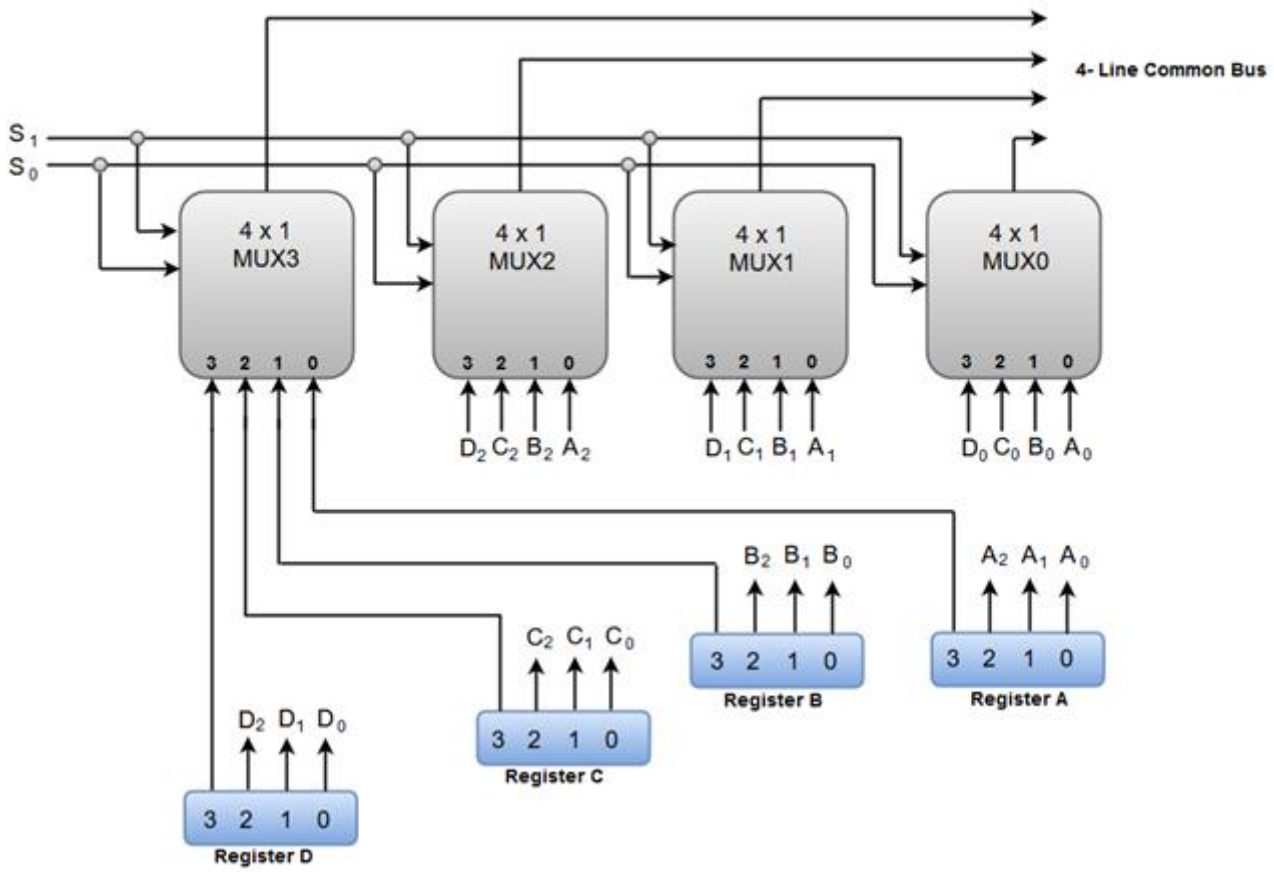
A bus structure, on the other hand, is more efficient for transferring information between registers in a multi-register configuration system.

A bus consists of a set of common lines, one for each bit of register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during a particular register transfer.

The following block diagram shows a Bus system for four registers. It is constructed with the help of four 4 * 1 Multiplexers each having four data inputs (0 through 3) and two selection inputs (S1 and S2)

We have used labels to make it more convenient for you to understand the input-output configuration of a Bus system for four registers. For instance, output 1 of register A is connected to input 0 of MUX1.

Bus System for 4 Registers:



The two selection lines S1 and S2 are connected to the selection inputs of all four multiplexers. The selection lines choose the four bits of one register and transfer them into the four-line common bus.

When both of the select lines are at low logic, i.e. S1S0 = 00, the 0 data inputs of all four multiplexers are selected and applied to the outputs that forms the bus. This, in turn, causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers.

Similarly, when S1S0 = 01, register B is selected, and the bus lines will receive the content provided by register B.

The following function table shows the register that is selected by the bus for each of the four possible binary values of the Selection lines.

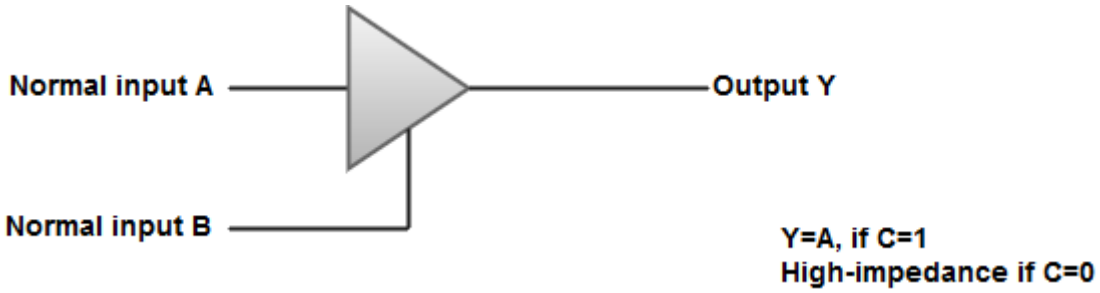
S1	S0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

A bus system can also be constructed using **three-state gates** instead of multiplexers.

The **three state gates** can be considered as a digital circuit that has three gates, two of which are signals equivalent to logic 1 and 0 as in a conventional gate. However, the third gate exhibits a high-impedance state.

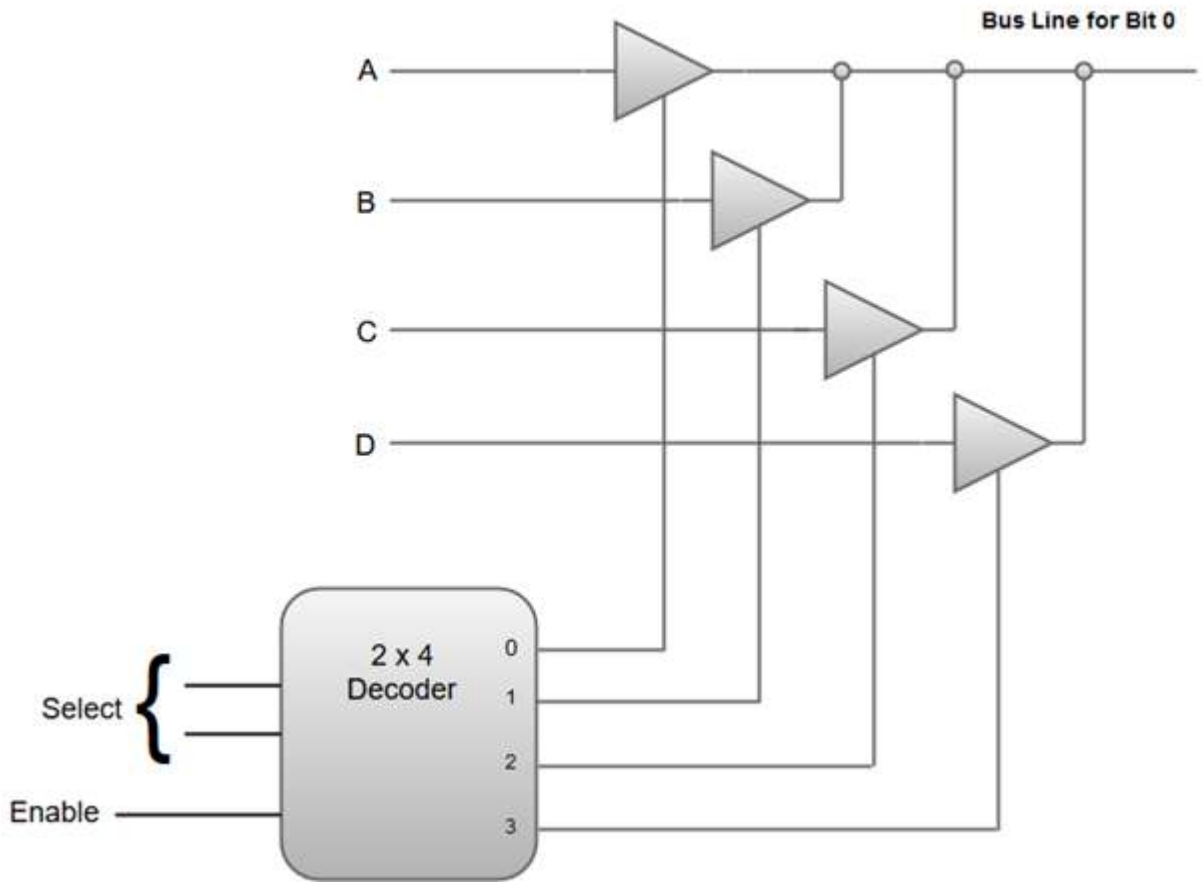
The most commonly used three state gates in case of the bus system is a **buffer gate**.

The graphical symbol of a three-state buffer gate can be represented as:



The following diagram demonstrates the construction of a bus system with three-state buffers.

Bus line with three state buffer:



- The outputs generated by the four buffers are connected to form a single bus line.
- Only one buffer can be in active state at a given point of time.
- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.
- A 2 * 4 decoder ensures that no more than one control input is active at any given point of time.

4) Explain about memory transfer?

Most of the standard notations used for specifying operations on memory transfer are stated below.

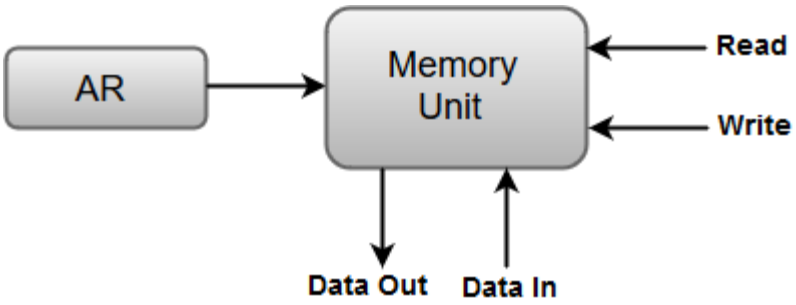
- The transfer of information from a memory unit to the user end is called a **Read** operation.
- The transfer of new information to be stored in the memory is called a **Write** operation.
- A memory word is designated by the letter **M**.
- We must specify the address of memory word while writing the memory transfer operations.
- The **address register** is designated by **AR** and the **data register** by **DR**.
- Thus, a read operation can be stated as:

1. Read: DR ← M[AR]

- The **Read** statement causes a transfer of information into the data register (DR) from the memory word (M) selected by the address register (AR).
- And the corresponding write operation can be stated as:

2.Write: M [AR] ← R1

- The Write statement causes a transfer of information from register R1 into the memory word (M) selected by address register (AR).



5) Explain about 4 bit binary adder?

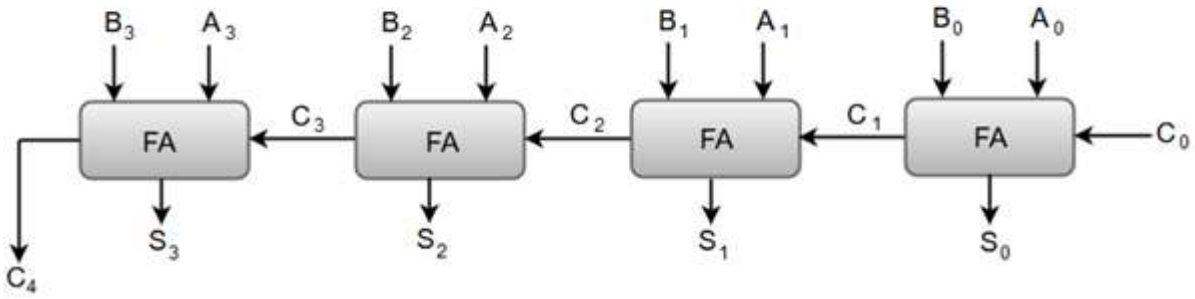
The Add micro-operation requires registers that can hold the data and the digital components that can perform the arithmetic addition.

A Binary Adder is a digital circuit that performs the arithmetic sum of two binary numbers provided with any length.

A Binary Adder is constructed using full-adder circuits connected in series, with the output carry from one full-adder connected to the input carry of the next full-adder.

The following block diagram shows the interconnections of four full-adder circuits to provide a 4-bit binary adder.

4 bit binary adder:



- The augend bits (A) and the addend bits (B) are designated by subscript numbers from right to left, with subscript '0' denoting the low-order bit.

- The carry inputs starts from C0 to C3 connected in a chain through the full-adders. C4 is the resultant output carry generated by the last full-adder circuit.
- The output carry from each full-adder is connected to the input carry of the next-high-order full-adder.
- The sum outputs (S0 to S3) generates the required arithmetic sum of augend and addend bits.
- The ***n*** data bits for the **A** and **B** inputs come from different source registers. For instance, data bits for **A** input comes from source register R1 and data bits for **B** input comes from source register R2.
- The arithmetic sum of the data inputs of A and B can be transferred to a third register or to one of the source registers (R1 or R2).

6) Give detail about Binary Adder-Subtractor?

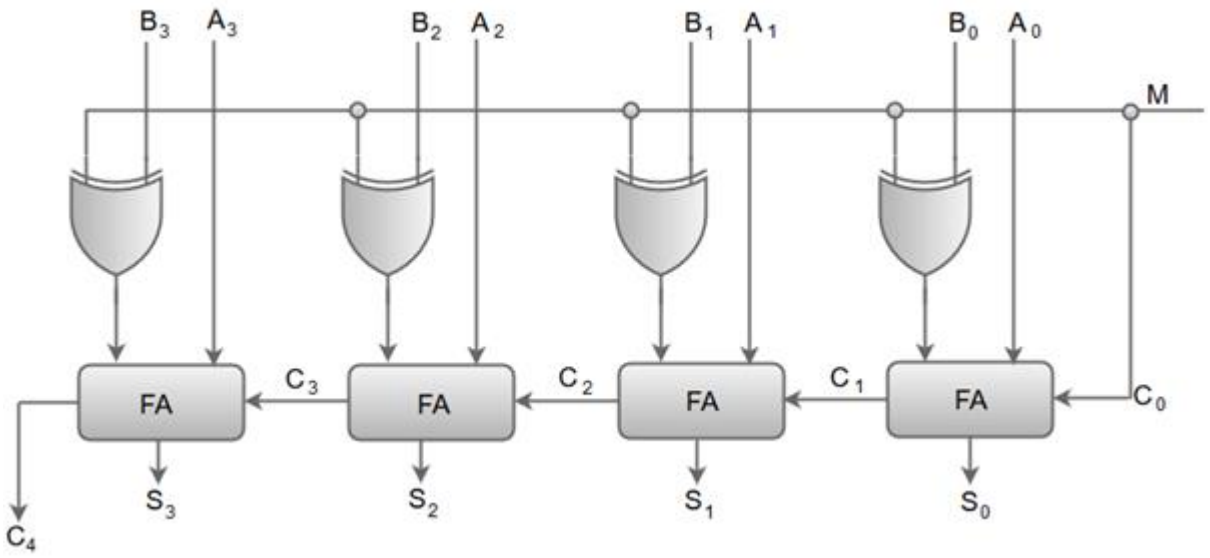
The Subtraction micro-operation can be done easily by taking the 2's compliment of addend bits and adding it to the augend bits.

Note: The 2's compliment can be obtained by taking the 1's compliment and adding one to the least significant pair of bits. The 1's compliment can be implemented with inverters, and one can be added to the sum through the input carry.

The Arithmetic micro-operations like addition and subtraction can be combined into one common circuit by including an exclusive-OR gate with each full adder.

The block diagram for a 4-bit adder-subtractor circuit can be represented as:

4 bit adder-subtractor:



- When the mode input (M) is at a low logic, i.e. '0', the circuit act as an adder and when the mode input is at a high logic, i.e. '1', the circuit act as a subtractor.
- The exclusive-OR gate connected in series receives input M and one of the inputs B.
- When M is at a low logic, we have B⊕ 0 = B. The full-adders receive the value of B, the input carry is 0, and the circuit performs A plus B.

- When M is at a high logic, we have B⊕ 1 = B' and C0 = 1. The B inputs are complemented, and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B.

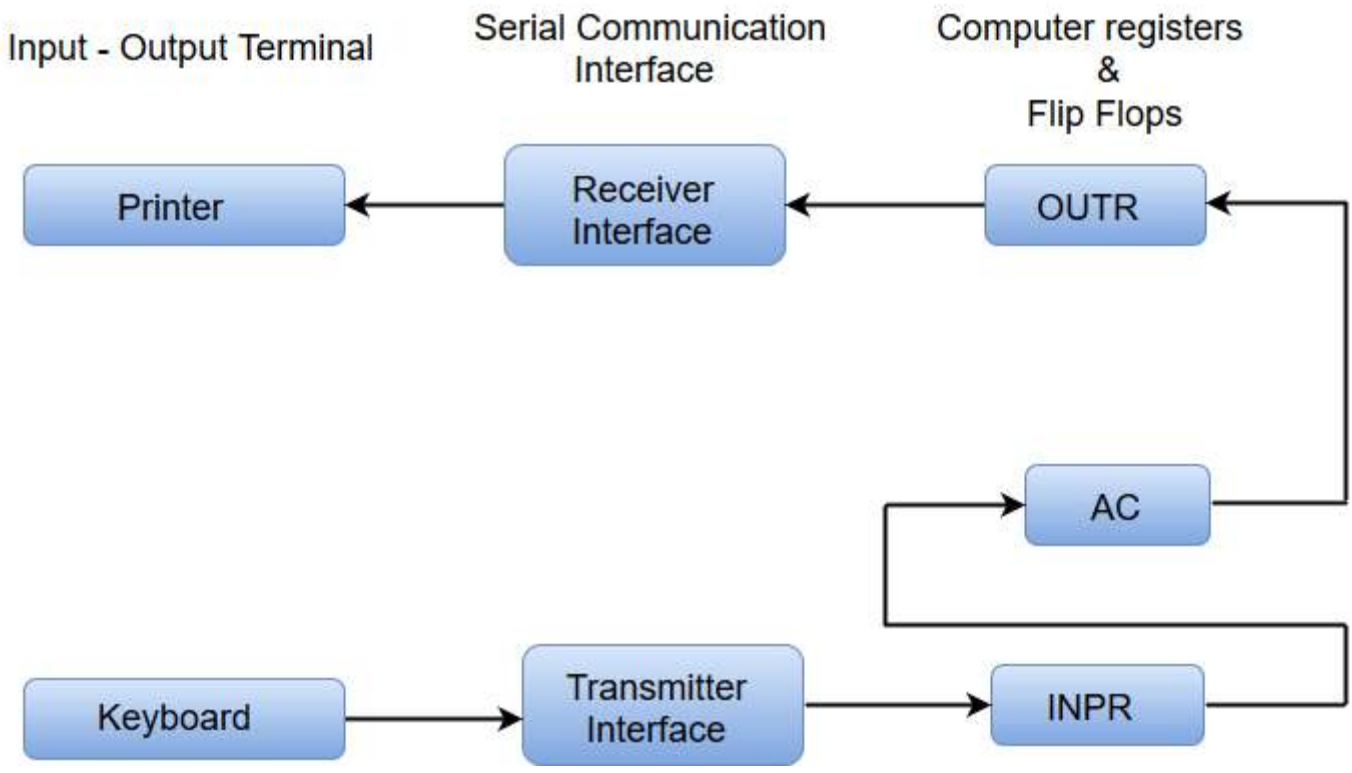
7) Describe about Input-Output Configuration?

In computer architecture, input-output devices act as an interface between the machine and the user.

Instructions and data stored in the memory must come from some input device. The results are displayed to the user through some output device.

The following block diagram shows the input-output configuration for a basic computer.

Input - Output Configuration:



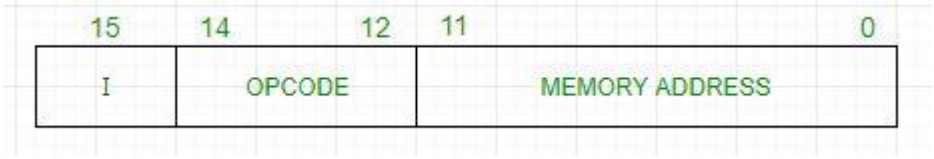
- The input-output terminals send and receive information.
- The amount of information transferred will always have eight bits of an alphanumeric code.
- The information generated through the keyboard is shifted into an input register 'INPR'.
- The information for the printer is stored in the output register 'OUTR'.
- Registers INPR and OUTR communicate with a communication interface serially and with the AC in parallel.
- The transmitter interface receives information from the keyboard and transmits it to INPR.

- The receiver interface receives information from OUTR and sends it to the printer serially.

8) Explain about memory reference instructions?

The basic computer has 16-bit instruction register (IR) which can denote either memory reference or register reference or input-output instruction.

1. **Memory Reference –** These instructions refer to memory address as an operand. The other operand is always accumulator. Specifies 12-bit address, 3-bit opcode (other than 111) and 1-bit addressing mode for direct and indirect addressing.

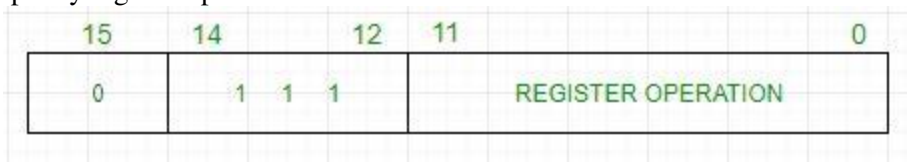


Example –
IR register contains = 0001XXXXXXXXXXX, i.e. ADD after fetching and decoding of instruction we find out that it is a memory reference instruction for ADD operation.

Hence, DR ← M[AR]

AC ← AC + DR, SC ← 0

2. **Register Reference –** These instructions perform operations on registers rather than memory addresses. The IR(14 – 12) is 111 (differentiates it from memory reference) and IR(15) is 0 (differentiates it from input/output instructions). The rest 12 bits specify register operation.



Example –
IR register contains = 0111001000000000, i.e. CMA after fetch and decode cycle we find out that it is a register reference instruction for complement accumulator.

Hence, AC ← ~AC

3. **Input/Output –** These instructions are for communication between computer and outside environment. The IR(14 – 12) is 111 (differentiates it from memory reference) and IR(15) is 1 (differentiates it from register reference instructions). The rest 12 bits specify I/O operation.

Example –
IR register contains = 1111100000000000, i.e. INP after fetch and decode cycle we find out that it is an input/output instruction for inputing character. Hence, INPUT character from peripheral device.
/li>

The set of instructions incorporated in16 bit IR register are:

1. **Arithmetic, logical and shift instructions (and, add, complement, circulate left, right, etc)**
2. **To move information to and from memory (store the accumulator, load the accumulator)**
3. **Program control instructions with status conditions (branch, skip)**
4. **Input output instructions (input character, output character)**

Symbol	Hexadecimal Code	Description

Symbol	Hexadecimal Code		Description
AND	0xxx	8xxx	And memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load memory word to AC
STA	3xxx	Bxxx	Store AC content in memory
BUN	4xxx	Cxxx	Branch Unconditionally
BSA	5xxx	Dxxx	Add memory word to AC
ISZ	6xxx	Exxx	Increment and skip if 0
CLA	7800		Clear AC
CLE	7400		Clear E(overflow bit)
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instruction if AC > 0
SNA	7008		Skip next instruction if AC < 0
SZA	7004		Skip next instruction if AC = 0
SE	7002		Skip next instruction if E = 0
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
IEN	F080		Interrupt On

9)Expain about ALSU?

Arithmetic Logic Shift Unit (ALSU) is a part of Arithmetic Logic Unit (ALU) of a computer system. It is a digital circuit that performs arithmetic calculations, logical and shift operations. Instead of having individual registers performing the microoperations directly, computer systems employ a number of storage registers connected to a common operational unit called an arithmetic logic unit, abbreviated ALU. ALSU consists of arithmetic, logical and shift circuits. In each stage, the circuit can perform 8 arithmetic operations, 16 logical operations and 2 shift operations.

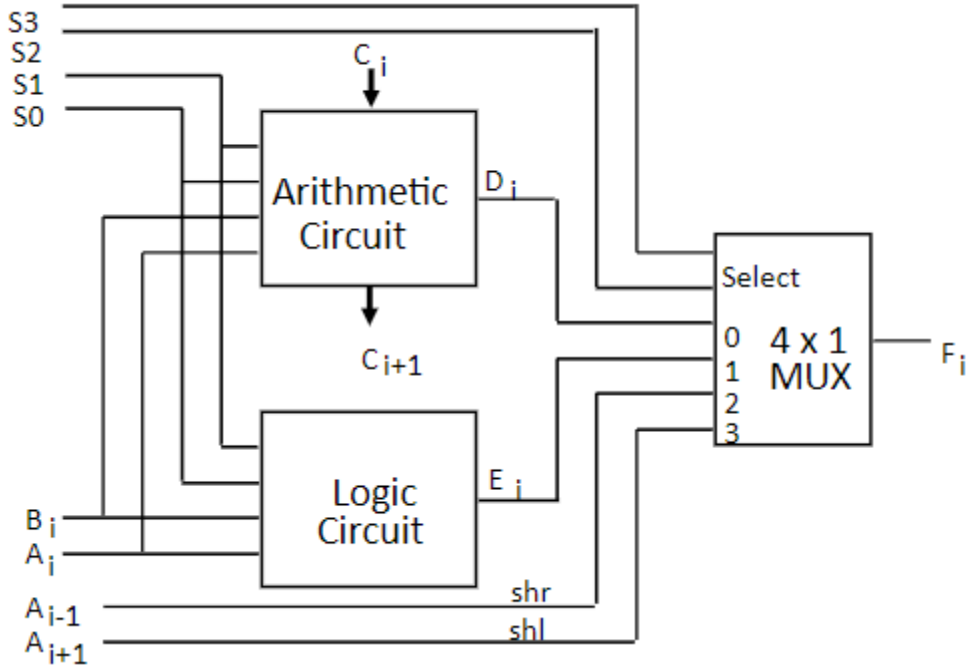


Figure 1 — Arithmetic Logic Shift Unit

Characteristics of the ALSU

1. Arithmetic logic shift unit in digital circuit performs all the logical and arithmetic operations.
2. The operations like addition, subtraction, multiplication and division are referred as Arithmetic operations.
3. The logical operations refer to operations on numbers and special character operations.

The ALSU is responsible for the conditions given below:

1. Equal to
2. Less than

3. Greater than

Functions of the ALSU

A particular microoperation is selected with inputs S1 and S0. A 4 x 1 multiplexer at the output chooses between an arithmetic output in Ei and a logic output in Hi. The data in the multiplexer are selected with inputs S3 and S2. The other two data inputs to the multiplexer receive inputs Ai — 1 for the shift-right operation and Ai + 1 for the shift-left operation.

Operation select						
S_3	S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	×	$F = A \wedge B$	AND
0	1	0	1	×	$F = A \vee B$	OR
0	1	1	0	×	$F = A \oplus B$	XOR
0	1	1	1	×	$F = \overline{A}$	Complement A
1	0	×	×	×	$F =$ shr A	Shift right A into F
1	1	×	×	×	$F =$ shl A	Shift left A into F

Table 1 — Functions of Arithmetic Logic Shift Unit

The circuit of Figure 1 must be repeated n times for an n-bit ALU. The output carry Ci + 1 of a given arithmetic stage must be connected to the input carry Ci of the next stage in sequence. The input carry to the first stage is the input carry Cin, which provides a selection variable for the arithmetic operations.

UNIT II

SHORT ANSWERS:

1) Define hardwired control?

Hardwired Control: When the control signals are generated by hardware using conventional logic design techniques, the control unit is said to be hardwired. The key characteristics are
High speed of operation
Expensive Relatively complex
No flexibility of adding new instructions .

2) Define microprogrammed control?

Control information is stored in control memory. Control memory is programmed to initiate the required sequence of micro-operations. The key characteristics are Speed of operation is low when compared with hardwired
Less complex
Less expensive
Flexibility to add new instructions

3)What is control word?

Control Word: The control variables at any given time can be represented by a string of 1's and 0's called a control word. All control words can be programmed to perform various operations on the components of the system

4)What is control variable?

The control function that specifies a microoperation is called as control variable. When control variable is in one binary state, the corresponding microoperation is executed. For the other¬ binary state the state of registers does not change. The active state of a control variable may be either 1 state or the 0 state, depending on the application.
For bus-organized systems the control signals that specify microoperations are groups of bits that select¬ the paths in multiplexers, decoders, and arithmetic logic unit.

5)Define subrountines?

Subroutines are programs that are used by other routines to accomplish a particular task and can be called from any point within the main body of the microprogram. Frequently many microprograms contain identical section of code.

6) Difference between Hardwired Control and Microprogrammed Control?

Hardwired Control	**Microprogrammed Control**
Technology is circuit based.	Technology is software based.
It is implemented through flip-flops, gates, decoders etc.	Microinstructions generate signals to control the execution of instructions.
Fixed instruction format.	Variable instruction format (16-64 bits per instruction).
Instructions are register based.	Instructions are not register based.
ROM is not used.	ROM is used.
It is used in RISC.	It is used in CISC.
Faster decoding.	Slower decoding.

Difficult to modify.	Easily modified.
Chip area is less.	Chip area is large.

7)What are the different types of addressing modes available?

The different types of addressing modes are:

1.Immediate addressing mode
2.Register addressing mode
3.Direct or absolute addressing mode
4.Indirect addressing mode
5.Indexed addressing mode
6.Relative addressing mode
7.Auto increment
8. Auto decrement

8)What is processor time of a program?

The period during which the processor is active is called processor time of a program. Itdepends on the hardware involved in the execution of individual machine instructions

9)Define Register addressing mode?

In register addressing mode, the operand is the contents of a processor register. The name(address) of the register is given in the instruction.Effective address (EA) = Ri, Where Ri is a processor register.

10)Define absolute addressing mode?

In absolute addressing mode, the operand is in a memory location. The addresses ofthis location are given explicitly in the instruction. This is also called as direct addressing mode.EA = Loc Where loc is the memory address.

LONG ANSWERS:

1)Describe about general register organization?

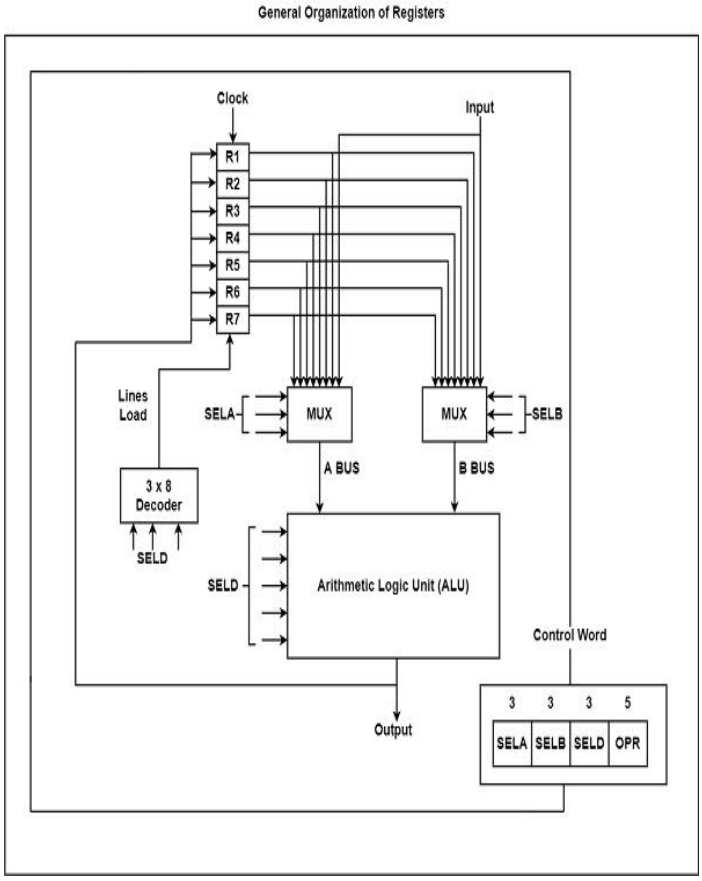
A set of flip-flops forms a register. A register is a unique high-speed storage area in the CPU. They include combinational circuits that implement data processing. The information is always defined in a register before processing. The registers speed up the implementation of programs.

Registers implement two important functions in the CPU operation are as follows −

- It can support a temporary storage location for data. This supports the directly implementing programs to have fast access to the data if required.
- It can save the status of the CPU and data about the directly implementing program.

Example − Address of the next program instruction, signals get from the external devices and error messages, and including different data is saved in the registers.

If a CPU includes some registers, therefore a common bus can link these registers. A general organization of seven CPU registers is displayed in the figure.



The CPU bus system is managed by the control unit. The control unit explicit the data flow through the ALU by choosing the function of the ALU and components of the system.

Consider R1 ← R2 + R3, the following are the functions implemented within the CPU −

MUX A Selector (SELA) − It can place R2 into bus A.

MUX B Selector (SELB) − It can place R3 into bus B.

ALU Operation Selector (OPR) − It can select the arithmetic addition (ADD).

Decoder Destination Selector (SELD) − It can transfers the result into R1.

The multiplexers of 3-state gates are performed with the buses. The state of 14 binary selection inputs determines the control word. The 14-bit control word defines a micro-operation.

The encoding of register selection fields is specified in the table.

Encoding of Register Selection Field

Binary Code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1

Binary Code	SELA	SELB	SELD
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

There are several micro-operations are implemented by the ALU. Few of the operations implemented by the ALU are displayed in the table.

Encoding of ALU Operations

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A + B	ADD
00101	Subtract A - B	SUB
00110	Decrement A	DECA
01000	ADD A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA

OPR Select	Operation	Symbol
10000	Shift right A	SHRA
11000	Shift left A	SHLA

There are some ALU micro-operations are shown in the table.

ALU Micro-Operations

Micro-operation	SELA	SELB	SELD	OPR	Control Word			
R1 ← R2 – R3	R2	R3	R1	SUB	010	011	001	00101
R4 ← R4 ∨ R5	R4	R5	R4	OR	100	101	100	01010
R6 ← R6 + R1	-	R6	R1	INCA	110	000	110	00001
R7 ← R1	R1	-	R7	TSFA	001	000	111	00000
Output ← R2	R2	–	None	TSFA	010	000	000	00000
Output ← Input	Input	-	None	TSFA	000	000	000	00000
R4 ← shl R4	R4	-	R4	SHLA	100	000	100	11000
R5 ← 0	R5	R5	R5	XOR	101	101	101	01100

2) Give detail about design of control unit?

The Control Unit is classified into two major categories:

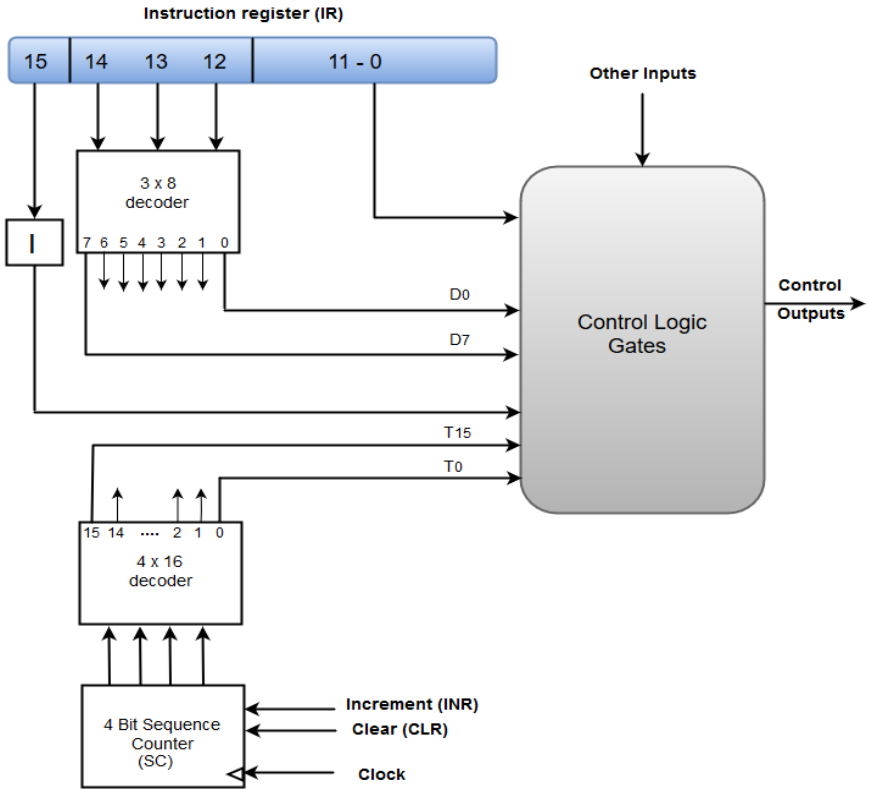
1. Hardwired Control
2. Microprogrammed Control

Hardwired Control

The Hardwired Control organization involves the control logic to be implemented with gates, flip-flops, decoders, and other digital circuits.

The following image shows the block diagram of a Hardwired Control organization.

Control Unit of a Basic Computer:



- A Hard-wired Control consists of two decoders, a sequence counter, and a number of logic gates.
- An instruction fetched from the memory unit is placed in the instruction register (IR).
- The component of an instruction register includes; I bit, the operation code, and bits 0 through 11.
- The operation code in bits 12 through 14 are coded with a 3 x 8 decoder.
- The outputs of the decoder are designated by the symbols D0 through D7.
- The operation code at bit 15 is transferred to a flip-flop designated by the symbol I.
- The operation codes from Bits 0 through 11 are applied to the control logic gates.
- The Sequence counter (SC) can count in binary from 0 through 15.

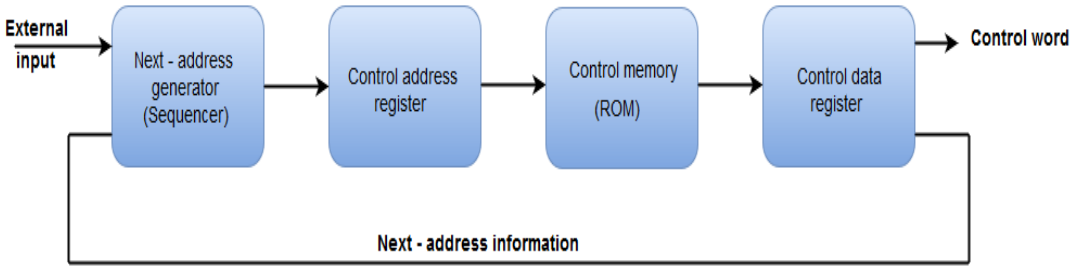
Micro-programmed Control

The Microprogrammed Control organization is implemented by using the programming approach.

In Microprogrammed Control, the micro-operations are performed by executing a program consisting of micro-instructions.

The following image shows the block diagram of a Microprogrammed Control organization.

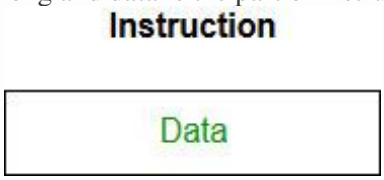
Microprogrammed Control Unit of a Basic Computer:



- The Control memory address register specifies the address of the micro-instruction.
- The Control memory is assumed to be a ROM, within which all control information is permanently stored.
- The control register holds the microinstruction fetched from the memory.
- The micro-instruction contains a control word that specifies one or more micro-operations for the data processor.
- While the micro-operations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.
- The next address generator is often referred to as a micro-program sequencer, as it determines the address sequence that is read from control memory.

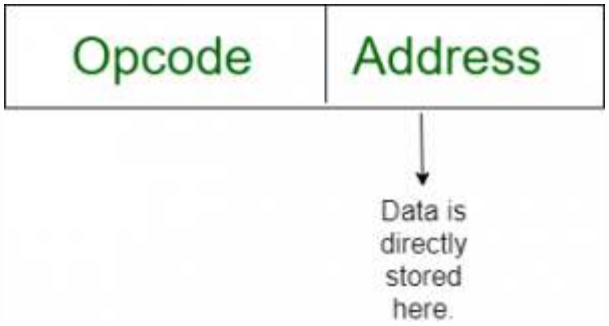
3) Explain about different types of addressing modes?

- **Implied mode:**: In implied addressing the operand is specified in the instruction itself. In this mode the data is 8 bits or 16 bits long and data is the part of instruction.Zero address instruction are designed with implied addressing mode.



Example: CLC (used to reset Carry flag to 0)

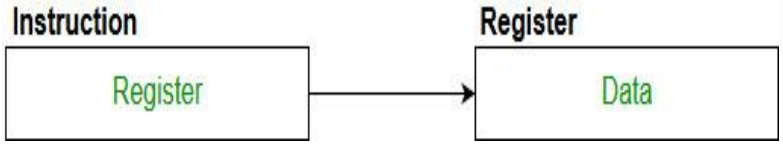
- **Immediate addressing mode (symbol #):**In this mode data is present in address field of instruction .Designed like one address instruction format.
Note: Limitation in the immediate mode is that the range of constants are restricted by size of address field.



Example: MOV AL, 35H (move the data 35H into AL register)

- **Register mode:** In register addressing the operand is placed in one of 8 bit or 16 bit general purpose registers. The data is in the register that is specified by the instruction.

Here one register reference is required to access the data.



Example: MOV AX,CX (move the contents of CX register to AX register)

- **Register Indirect mode**: In this addressing the operand's offset is placed in any one of the registers BX,BP,SI,DI as specified in the instruction. The effective address of the data is in the base register or an index register that is specified by the instruction.
Here two register reference is required to access the data.

The 8086 CPUs let you access memory indirectly through a register using the register indirect addressing modes.

- MOV AX, [BX](move the contents of memory location s

addressed by the register BX to the register AX)

- **Auto Indexed (increment mode)**: Effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next consecutive memory location.**(R1)+**.
Here one register reference,one memory reference and one ALU operation is required to access the data.
Example:

```
Add R1, (R2)+  // OR
R1 = R1 +M[R2]
R2 = R2 + d
```

Useful for stepping through arrays in a loop. R2 – start of array d – size of an element

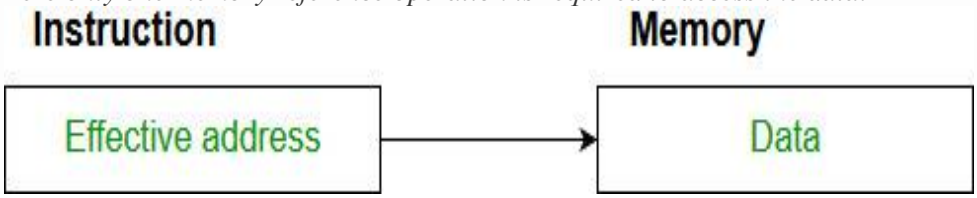
- **Auto indexed (decrement mode)**: Effective address of the operand is the contents of a register specified in the instruction. Before accessing the operand, the contents of this register are automatically decremented to point to the previous consecutive memory location. –**(R1)**
Here one register reference,one memory reference and one ALU operation is required to access the data.

Example:

```
Add R1,-(R2)   //OR
R2 = R2-d
R1 = R1 + M[R2]
```

Auto decrement mode is same as auto increment mode. Both can also be used to implement a stack as push and pop . Auto increment and Auto decrement modes are useful for implementing "Last-In-First-Out" data structures.

- **Direct addressing/ Absolute addressing Mode (symbol [ ]):** The operand's offset is given in the instruction as an 8 bit or 16 bit displacement element. In this addressing mode the 16 bit effective address of the data is the part of the instruction.
Here only one memory reference operation is required to access the data.



Example:ADD AL,[0301] //add the contents of offset address 0301 to AL

- **Indirect addressing Mode (symbol @ or ())**:In this mode address field of instruction contains the address of effective address.Here two references are required.
  1st reference to get effective address.
  2nd reference to access the data.
  Based on the availability of Effective address, Indirect mode is of two kind:

1. Register Indirect:In this mode effective address is in the register, and corresponding register name will be maintained in the address field of an instruction.
   Here one register reference,one memory reference is required to access the data.
2. Memory Indirect:In this mode effective address is in the memory, and corresponding memory address will be maintained in the address field of an instruction.
   Here two memory reference is required to access the data.

- **Indexed addressing mode**: The operand's offset is the sum of the content of an index register SI or DI and an 8 bit or 16 bit displacement.

Example:MOV AX, [SI +05]

- **Based Indexed Addressing:** The operand's offset is sum of the content of a base register BX or BP and an index register SI or DI.

Example: ADD AX, [BX+SI]

Based on Transfer of control, addressing modes are:

- **PC relative addressing mode:** PC relative addressing mode is used to implement intra segment transfer of control, In this mode effective address is obtained by adding displacement to PC.
- EA= PC + Address field value

PC= PC + Relative value.

- **Base register addressing mode:**Base register addressing mode is used to implement inter segment transfer of control.In this mode effective address is obtained by adding base register value to address field value.
- EA= Base register + Address field value.
- PC= Base register + Relative value.

Note:

1. PC relative nad based register both addressing modes are suitable for program relocation at runtime.
2. Based register addressing mode is best suitable to write position independent codes.

4)Explain about instruction format?

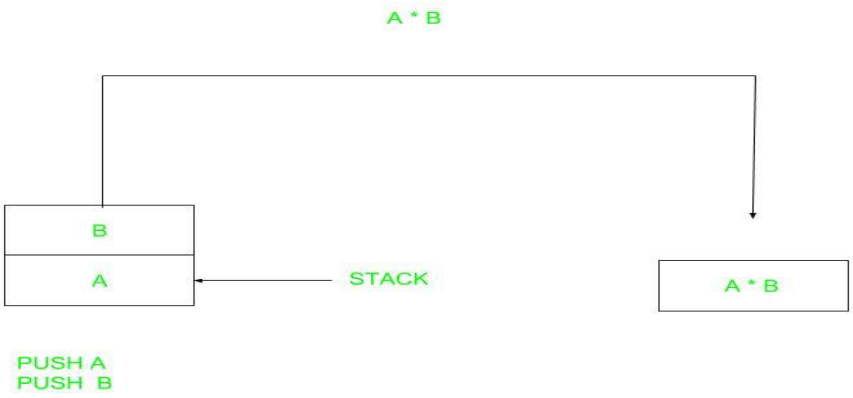
1. Single Accumulator organization
2. General register organization
3. Stack organization

In the first organization, the operation is done involving a special register called the accumulator. In second on multiple registers are used for the computation purpose. In the third organization the work on stack basis operation due to which it does not contain any address field. Only a single organization doesn't need to be applied, a blend of various organizations is mostly what we see generally.

Based on the number of address, instructions are classified as:

Note that we will use X = (A+B)*(C+D) expression to showcase the procedure.

1. **Zero Address Instructions –**



A stack-based computer does not use the address field in the instruction. To evaluate an expression first it is converted to reverse Polish Notation i.e. Postfix Notation.

Expression: X = (A+B)*(C+D)

Postfixed : X = AB+CD+*

TOP means top of stack

M[X] is any memory location

PUSH A TOP = A

PUSH B TOP = B

ADD TOP = A+B

PUSH C TOP = C

PUSH D TOP = D

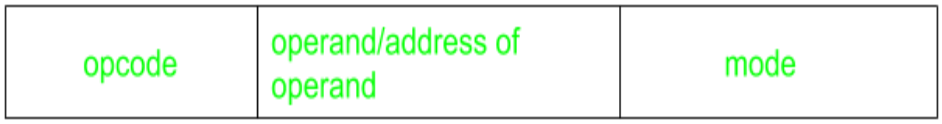
ADD TOP = C+D

MUL TOP = (C+D)*(A+B)

POP X M[X] = TOP

2 .One Address Instructions –

This uses an implied ACCUMULATOR register for data manipulation. One operand is in the accumulator and the other is in the register or memory location. Implied means that the CPU already knows that one operand is in the accumulator so there is no need to specify it.



Expression: X = (A+B)*(C+D)

AC is accumulator

M[] is any memory location

M[T] is temporary location

LOAD A AC = M[A]

ADD B AC = AC + M[B]

STORE T M[T] = AC

LOAD C AC = M[C]

ADD D AC = AC + M[D]

MUL T AC = AC * M[T]

STORE X M[X] = AC

3.Two Address Instructions –
This is common in commercial computers. Here two addresses can be specified in the instruction. Unlike earlier in one address instruction, the result was stored in the accumulator, here the result can be stored at different locations rather than just accumulators, but require more number of bit to represent address.

Here destination address can also contain operand.

Expression: X = (A+B)*(C+D)

R1, R2 are registers

M[] is any memory location

MOV R1, A R1 = M[A]

ADD R1, B R1 = R1 + M[B]

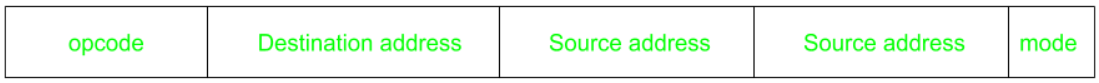
MOV R2, C R2 = C

ADD R2, D R2 = R2 + D

MUL R1, R2 R1 = R1 * R2

MOV X, R1 M[X] = R1

4.Three Address Instructions –
This has three address field to specify a register or a memory location. Program created are much short in size but number of bits per instruction increase. These instructions make creation of program much easier but it does not mean that program will run much faster because now instruction only contain more information but each micro operation (changing content of register, loading address in address bus etc.) will be performed in one cycle only.



Expression: X = (A+B)*(C+D)

R1, R2 are registers

M[] is any memory location

ADD R1, A, B R1 = M[A] + M[B]

ADD R2, C, D R2 = M[C] + M[D]

MUL X, R1, R2 M[X] = R1 * R2

5)Explain about program control?

Instructions of the computer are always stored in consecutive memory locations. These instructions are fetched from successive memory locations for processing and executing.

When an instruction is fetched from the memory, the program counter is incremented by 1 so that it points to the address of the next consecutive instruction in the memory. Once a data transfer and data manipulation instruction are executed, the program control along with the program counter, which holds the address of the next instruction to be fetched, is returned to the fetch cycle.

Data transfer and manipulation instructions specify the conditions for data processing operations, whereas the program control instructions specify the conditions that can alter the content of the program counter.

The change in the content of the program counter can cause an interrupt/break in the instruction execution. However, the program control instructions control the flow of program execution and are capable of branching to different program segments.

Some of the program control instructions are listed in the table.

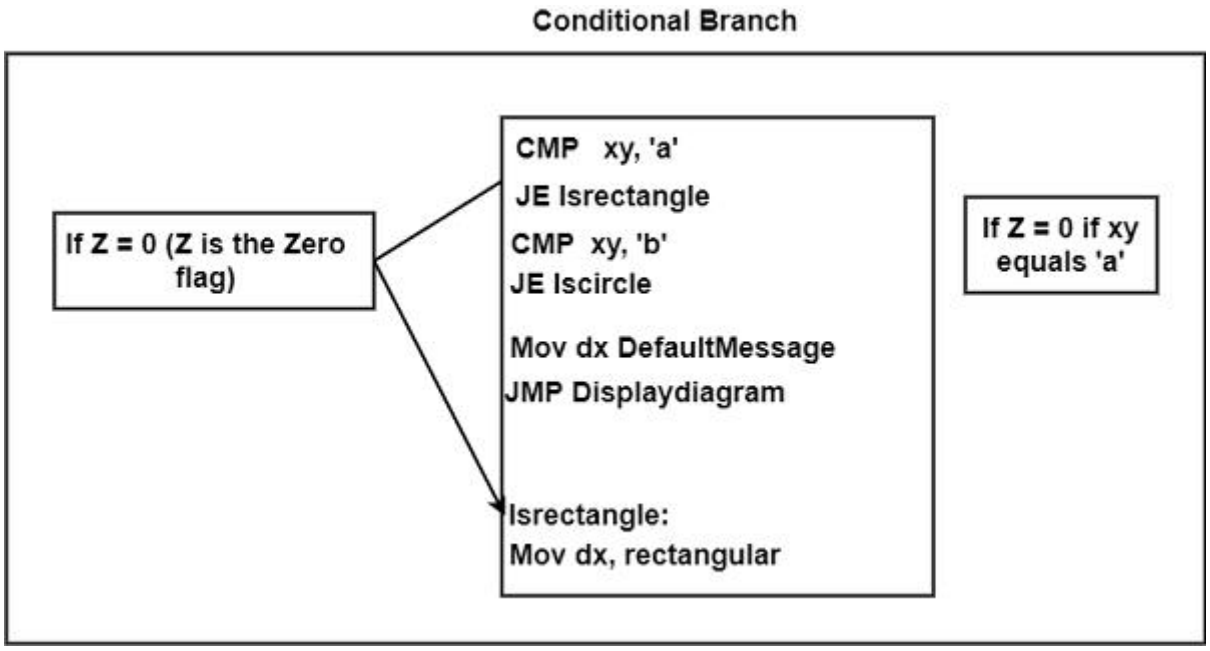
Program Control Instructions

Name	Mnemonics
Branch	BR
Jump	JMP
Skip	SKP
Call	Call
Return	RET
Compare (by Subtraction)	CMP
Test (by ANDing)	TST

The branch is a one-address instruction. It is represented as BR ADR, where ADR is a mnemonic for an address. The branch instruction transfers the value of ADR into the program counter. The branch and jump instructions are interchangeably used to mean the same. However, sometimes they denote different addressing modes.

The conditional branch instructions such as ‘branch if positive’, or ‘branch if zero’ specifies the condition to transfer the flow of execution. When the condition is met, the branch address is loaded in the program counter.

The figure depicts the conditional branch instructions.



The compare instruction performs an arithmetic subtraction. Here, the result of the operation is not saved; instead, the status bit conditions are set. The test instruction performs the logical AND operation on two operands and updates the status bits.

6)Give a detail about address sequence?

Microinstructions are saved in control memory in groups. These groups describe routines. Each computer instruction has its microprogram routine that can create micro-operations. These micro-operations can execute instructions. The hardware consists of controls for the address sequencing of the microinstructions of a similar routine. They also branch the microinstructions.

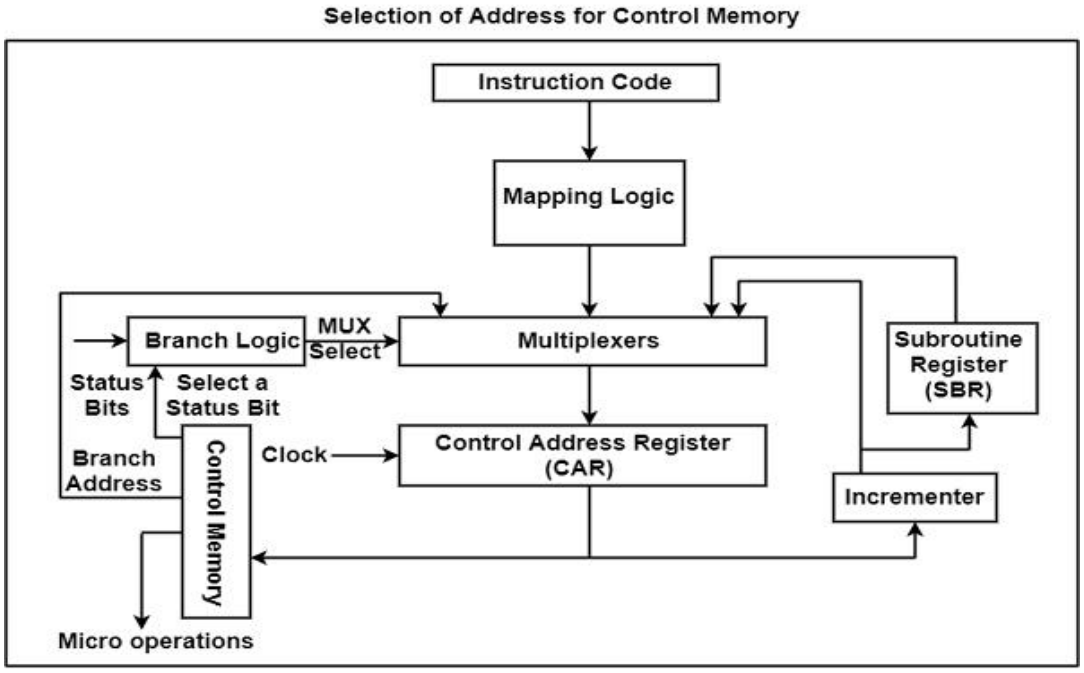
There are the following phases that the control has while implementing a computer instruction −

- When power is turned on, and address is initially loaded into the control address register. (This is the address of the first microinstruction).
- The control address register is incremented resulting in sequencing the fetch routine.
- After the fetch routine, the instruction is present in the IR of the computer.
- Next, the control memory retrieves the effective address of the operand from the routine.
- Thus, the mapping process appears from the instruction bits to a control memory address.
- It depends on the opcodes of instruction the microinstructions of the processor registers are generated. Each of these microinstructions has a separate microprogram routine stored.
- The instruction code bits are changed into the address where the routine is placed and is known as the mapping process. A mapping process transforms the microinstruction into a control memory address.
- Next, subroutines are called and processes are returned.
- After the completion of the routine, the control address register is incremented to sequence the instruction is implemented. It can also be based on the values of status bits in processor registers. External registers are needed through microprograms to save return addresses that use subroutines. After the instruction is performed, the control returns to the fetch routine. This is done by branching the microinstruction to the first address in the fetch routine.

The diagram shows the block diagram of a control memory and its associated hardware to support in choosing the next microinstruction. The microinstruction present in the control memory has a set of bits that facilitate to start off the micro-operations in registers.

There are four different directions are showed in the figure from where the control address register recovers its address. The CAR is incremented by the incrementer and selects the next instruction. In multiple fields of microinstruction, the branching address can be determined to result in branching.

It can specify the condition of the status bits of microinstruction, conditional branching can be applied. A mapping logic circuit can share an external address. A special register can save the return address so that when the microprogram needs to return from the subroutine, it can need the value from the unique register.



7)Describe about data and manipulation instruction?

Data Manipulation Instructions :

Data manipulation instructions perform operations on data and provide the computational capabilities for the computer. The data manipulation instructions in a typical computer usually divided into three basic types as follows.

1. Arithmetic instructions
2. Logical and bit manipulation instructions
3. Shift instructions

Arithmetic instructions :

The four basic arithmetic operations are addition, subtraction, multiplication, and division. Most computers provide instructions for all four operations.

Typical Arithmetic Instructions –

Name	Mnemonic	Example	Explanation
Increment	INC	INC B	It will increment the register B by 1 B<-B+1
Decrement	DEC	DEC B	It will decrement the register B by 1 B<-B-1
Add	ADD	ADD B	It will add contents of register B to the contents of the accumulator and store the result in the accumulator

			AC<-AC+B
Subtract	SUB	SUB B	It will subtract the contents of register B from the contents of the accumulator and store the result in the accumulator AC<-AC-B
Multiply	MUL	MUL B	It will multiply the contents of register B with the contents of the accumulator and store the result in the accumulator AC<-AC*B
Divide	DIV	DIV B	It will divide the contents of register B with the contents of the accumulator and store the quotient in the accumulator AC<-AC/B
Add with carry	ADDC	ADDC B	It will add the contents of register B and the carry flag with the contents of the accumulator and store the result in the accumulator AC<-AC+B+Carry flag
Subtract with borrow	SUBB	SUBB B	It will subtract the contents of register B and the carry flag from the contents of the accumulator and store the result in the accumulator AC<-AC-B-Carry flag
Negate(2's complement)	NEG	NEG B	It will negate a value by finding 2's complement of its single operand. This means simply operand by -1. B<-B'+1

1. **Logical and Bit Manipulation Instructions :**
Logical instructions perform binary operations on strings of bits stored in registers. They are useful for manipulating individual bits or a group of bits.
Typical Logical and Bit Manipulation Instructions –

Name	Mnemonic	Example	Explanation
Clear	CLR	CLR	It will set the accumulator to 0

			AC<-0
Complement	COM	COM A	It will complement the accumulator AC<-(AC)’
AND	AND	AND B	It will AND the contents of register B with the contents of accumulator and store it in the accumulator AC<-AC AND B
OR	OR	OR B	It will OR the contents of register B with the contents of accumulator and store it in the accumulator AC<-AC OR B
Exclusive-OR	XOR	XOR B	It will XOR the contents of register B with the contents of the accumulator and store it in the accumulator AC<-AC XOR B
Clear carry	CLRC	CLRC	It will set the carry flag to 0 Carry flag<-0
Set carry	SETC	SETC	It will set the carry flag to 1 Carry flag<-1
Complement carry	COMC	COMC	It will complement the carry flag Carry flag<- (Carry flag)’
Enable interrupt	EI	EI	It will enable the interrupt
Disable interrupt	DI	DI	It will disable the interrupt

2. **Shift Instructions :**
Shifts are operations in which the bits of a word are moved to the left or right. Shift instructions may specify either logical shifts, arithmetic shifts, or rotate-type operations.
Typical Shift Instructions –
Name Mnemonic

Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

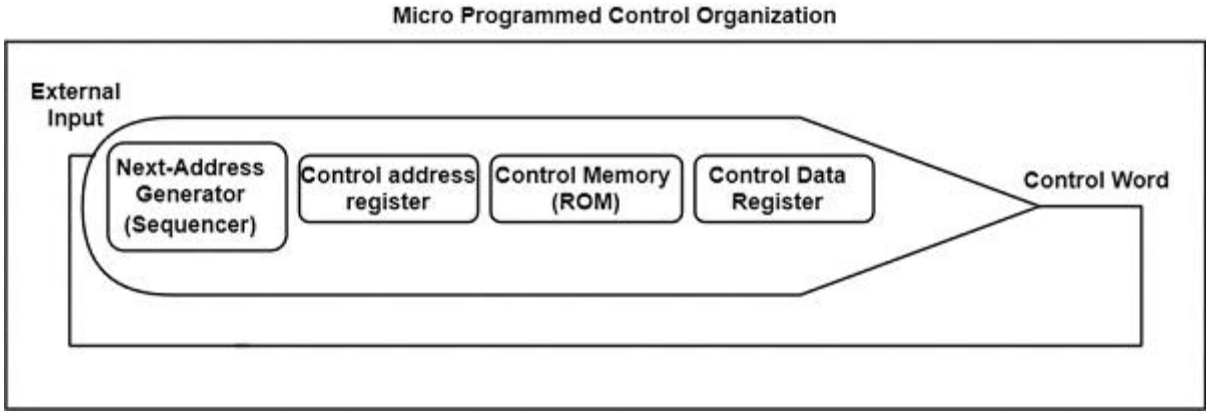
8)Expalin about organization of micro programmed control unit?

A control unit whose binary control values are saved as words in memory is called a microprogrammed control unit.

A controller results in the instructions to be implemented by constructing a definite collection of signals at each system clock beat. Each of these output signals generates one micro-operation including register transfer. Thus, the sets of control signals are generated definite micro-operations that can be saved in the memory.

Each bit that forms the microinstruction is linked to one control signal. When the bit is set, the control signal is active. When it is cleared the control signal turns inactive. These microinstructions in a sequence can be saved in the internal 'control' memory. The control unit of a microprogram-controlled computer is a computer inside a computer.

The following image shows the block diagram of a Microprogrammed Control organization.



There are the following steps followed by the microprogrammed control are −

- It can execute any instruction. The CPU should divide it down into a set of sequential operations. This set of operations are called microinstruction. The sequential micro-operations need the control signals to execute.
- Control signals saved in the ROM are created to execute the instructions on the data direction. These control signals can control the micro-operations concerned with a microinstruction that is to be performed at any time step.
- The address of the microinstruction is executed next is generated.
- The previous 2 steps are copied until all the microinstructions associated with the instruction in the set are executed.

The address that is supported to the control ROM originates from the micro counter register. The micro counter received its inputs from a multiplexer that chooses the output of an address ROM, a current address incrementer, and an address that is saved in the next address field of the current microinstruction.

Advantages of Microprogrammed Control Unit

There are the following advantages of microprogrammed control are as follows −

- It can more systematic design of the control unit.
- It is simpler to debug and change.
- It can retain the underlying structure of the control function.
- It can make the design of the control unit much simpler. Hence, it is inexpensive and less error-prone.
- It can orderly and systematic design process.
- It is used to control functions implemented in software and not hardware.
- It is more flexible.
- It is used to complex function is carried out easily.

Disadvantages of Microprogrammed Control Unit

There are the following disadvantages of microprogrammed control are as follows −

- Adaptability is obtained at more cost.
- It is slower than a hardwired control unit.

UNIT III

SHORT ANSWERS:

1)What is number system?

To a computer, everything is a number, i.e., alphabets, pictures, sounds, etc., are numbers. Number system is categorized into four types −

- Binary number system consists of only two values, either 0 or 1
- Octal number system represents values in 8 digits.
- Decimal number system represents values in 10 digits.
- Hexadecimal number system represents values in 16 digits.

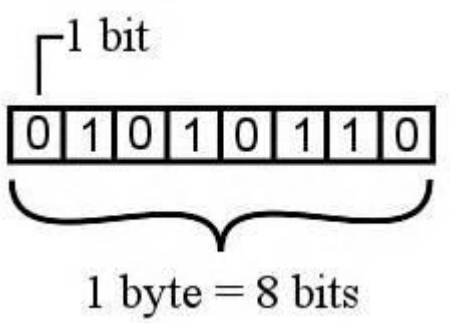
Number System		
System	**Base**	**Digits**
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9

Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

2) Define bits and bytes?

Bits − A bit is a smallest possible unit of data that a computer can recognize or use. Computer usually uses bits in groups.

Bytes − group of eight bits is called a byte. Half a byte is called a nibble.



3) Describe about fixed point representation?

Digital Computers use Binary number system to represent all types of information inside the computers. Alphanumeric characters are represented using binary bits (i.e., 0 and 1). Digital representations are easier to design, storage is easy, accuracy and precision are greater.

4)What are the four consecutive ?

The algorithm can be divided into four consecutive parts:

1. Check for zeros.
2. Align the mantissas.
3. Add or subtract the mantissas
4. Normalize the result

5) Define Magnetic Disk?

The gramophone record, which is circular like a disk and coated with magnetic material. Magnetic disks used in computer are made on the same principle. It rotates with very highspeed inside the computer drive.

6)What are the optical disks divided into?

Optical diskscan be divided into the following categories:

1. Compact Disk/ Read Only Memory (CD-ROM
2. Write Once, Read Many (WORM)
3. Erasable Optical Disk

7)Define CAM?

The time required to find an item stored in memory can be reduced considerably if stored data canbe identified for access by the content of the data itself rather than by an address.

A memory unit accessed by content is called an associative memory or content addressablememory (CAM).

8) What is cache memory?

Cache is a fast small capacity memory that should hold those information which are mostlikely to be accessed.
The basic operation of the cache is, when the CPU needs to access memory, the cache isexamined.

9)Define write through method?

The simplest and most commonly used procedure is to update main memory with every memorywrite operation.
The cache memory being updated in parallel if it contains the word at the specified address. Thisis called the *write-through* method.

10)Define virtual memory?

Virtual memory is used to give programmers the illusion that they have a very large memory attheir disposal, even though the computer actually has a relatively small main memory.A virtual memory system provides a mechanism for translating program-generated addresses intocorrect main memory locations.

LONG ANSWERS:

1)Explain about decimal arithmetic operation?

Decimal arithmetic operations refer to a digital function that does decimal micro-operations. This function adds or subtracts decimal numbers by forming 9's or 10's complement of the subtrahend. This decimal arithmetic unit first accepts coded decimal numbers and then generates output in the binary form.

Algorithms that are used for arithmetic operations with decimal data and binary data are alike. If the micro-operations symbol is interpreted correctly the same flowchart can be used for both multiplication and division.

The decimal numbers in BCD are stored in groups of four bits in the computer registers. When performing decimal micro-operations, every 4-bit group represents a decimal digit and has to be taken as a group

The table shows symbols for decimal arithmetic micro-operations.

Symbols for Decimal Arithmetic Micro-Operations

Symbolic Representation	Meaning
X ← X + Y	It can add decimal numbers and transfers the output to X.
Y′	9's complement of Y.
X ← X + Y′ + 1	It can add the content of X and 10's complement of Y and transfers the output to X.
dshr X	It can shifts the decimal number one digit towards the right in register

Symbolic Representation	Meaning
	X.
dshl X	It can shifts the decimal number one digit towards left in register X

In this table, we can see a bar over the symbol for the register letter. This refers to the 9's complement of decimal number that is stored in the register. When 1 is added to the 9's complement the 10's complement is produced.

Therefore, the symbol X ← X+ Y + 1 for decimal digits denotes, transfer of decimal sum that was formed by adding the original content X to the 10's complement of Y.

It may be confusing to use similar symbols for 9's complement and 1's complement in case both types of data are used in the same system.

Therefore, it would be better to implement a different symbol for the 9's complement. In case only one type of data is taken into consideration, the symbol would apply to the type of data used.

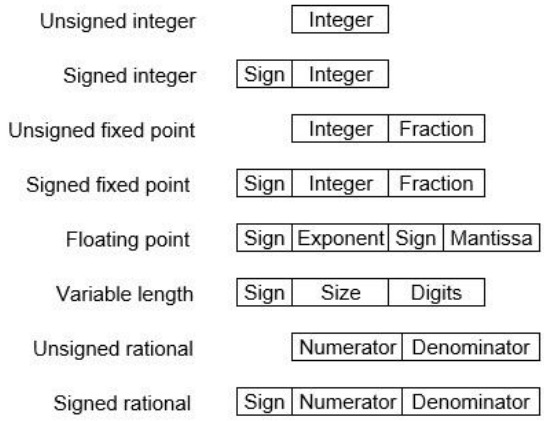
2) Explain Fixed point representation?

Digital Computers use Binary number system to represent all types of information inside the computers. Alphanumeric characters are represented using binary bits (i.e., 0 and 1). Digital representations are easier to design, storage is easy, accuracy and precision are greater.

There are various types of number representation techniques for digital number representation, for example: Binary number system, octal number system, decimal number system, and hexadecimal number system etc. But Binary number system is most relevant and popular for representing numbers in digital computer system.

Storing Real Number

These are structures as following below −



There are two major approaches to store real numbers (i.e., numbers with fractional component) in modern computing. These are (i) Fixed Point Notation and (ii) Floating Point Notation. In fixed point notation, there are a fixed number of digits after the decimal point, whereas floating point number allows for a varying number of digits after the decimal point.

Fixed-Point Representation −

This representation has fixed number of bits for integer part and for fractional part. For example, if given fixed-point representation is IIII.FFFF, then you can store minimum value is 0000.0001 and maximum value is 9999.9999. There are three parts of a fixed-point number representation: the sign field, integer field, and fractional field.

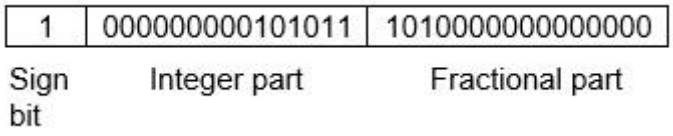
We can represent these numbers using:

- Signed representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 1's complement representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 2's complementation representation: range from $-(2^{(k-1)})$ to $(2^{(k-1)}-1)$, for k bits.

2's complementation representation is preferred in computer system because of unambiguous property and easier for arithmetic operations.

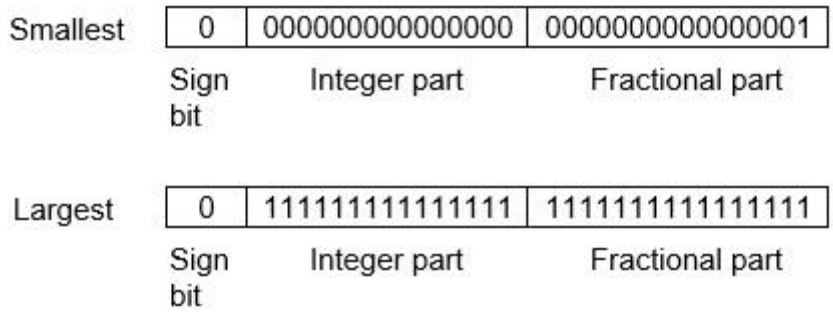
Example −Assume number is using 32-bit format which reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part.

Then, -43.625 is represented as following:



Where, 0 is used to represent + and 1 is used to represent. 000000000101011 is 15 bit binary value for decimal 43 and 1010000000000000 is 16 bit binary value for fractional 0.625.

The advantage of using a fixed-point representation is performance and disadvantage is relatively limited range of values that they can represent. So, it is usually inadequate for numerical analysis as it does not allow enough numbers and accuracy. A number whose representation exceeds 32 bits would have to be stored inexactly.



These are above smallest positive number and largest positive number which can be store in 32-bit representation as given above format. Therefore, the smallest positive number is $2^{-16} \approx 0.000015$ approximate and the largest positive number is $(2^{15}-1)+(1-2^{-16})=2^{15}(1-2^{-16}) =32768$, and gap between these numbers is 2^{-16}.

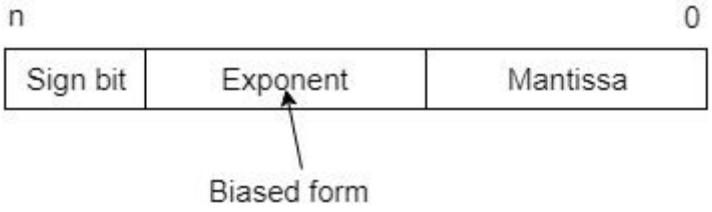
We can move the radix point either left or right with the help of only integer field is 1.

Floating-Point Representation −

This representation does not reserve a specific number of bits for the integer part or the fractional part. Instead it reserves a certain number of bits for the number (called the mantissa or significand) and a certain number of bits to say where within that number the decimal place sits (called the exponent).

The floating number representation of a number has two part: the first part represents a signed fixed point number called mantissa. The second part of designates the position of the decimal (or binary) point and is called the exponent. The fixed point mantissa may be fraction or an integer. Floating -point is always interpreted to represent a number in the following form: Mxr^e.

Only the mantissa m and the exponent e are physically represented in the register (including their sign). A floating-point binary number is represented in a similar manner except that is uses base 2 for the exponent. A floating-point number is said to be normalized if the most significant digit of the mantissa is 1.



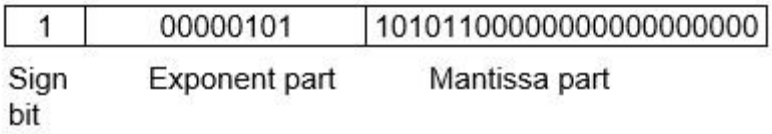
So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number.

Note that signed integers and exponent are represented by either sign representation, or one's complement representation, or two's complement representation.

The floating point representation is more flexible. Any non-zero number can be represented in the normalized form of $\pm(1.b_1b_2b_3 ...)_2x2^n$ This is normalized form of a number x.

Example −Suppose number is using 32-bit format: the 1 bit sign bit, 8 bits for signed exponent, and 23 bits for the fractional part. The leading bit 1 is not stored (as it is always 1 for a normalized number) and is referred to as a *"hidden bit*".

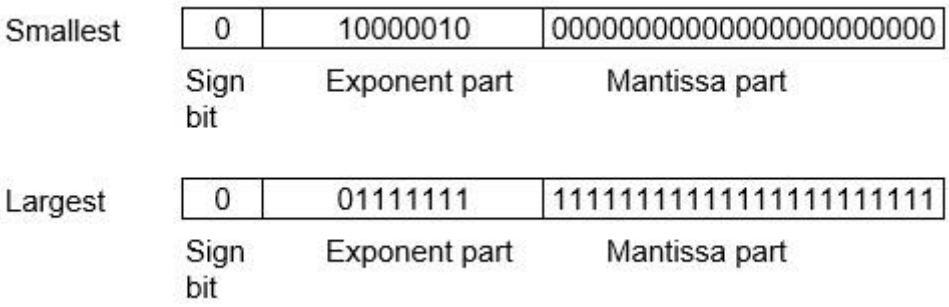
Then −53.5 is normalized as $-53.5=(-110101.1)_2=(-1.101011)x2^5$, which is represented as following below,



Where 00000101 is the 8-bit binary value of exponent value +5.

Note that 8-bit exponent field is used to store integer exponents $-126 \leq n \leq 127$.

The smallest normalized positive number that fits into 32 bits is $(1.00000000000000000000000)_2x2^{-126}=2^{-126}\approx1.18x10^{-38}$, and largest normalized positive number that fits into 32 bits is $(1.11111111111111111111111)_2x2^{127}=(2^{24}-1)x2^{104} \approx 3.40x10^{38}$. These numbers are represented as following below,



The precision of a floating-point format is the number of positions reserved for binary digits plus one (for the hidden bit). In the examples considered here the precision is 23+1=24.

The gap between 1 and the next normalized floating-point number is known as machine epsilon. the gap is $(1+2^{-23})-1=2^{-23}$for above example, but this is same as the smallest positive floating-point number because of non-uniform spacing unlike in the fixed-point scenario.

Note that non-terminating binary numbers can be represented in floating point representation, e.g., 1/3 = $(0.010101 ...)_2$ cannot be a floating-point number as its binary representation is non-terminating.

IEEE Floating point Number Representation −

IEEE (Institute of Electrical and Electronics Engineers) has standardized Floating-Point Representation as following diagram.

So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number. The sign bit is 0 for positive number and 1 for negative number. Exponents are represented by or two's complement representation.

According to IEEE 754 standard, the floating-point number is represented in following ways:

- Half Precision (16 bit): 1 sign bit, 5 bit exponent, and 10 bit mantissa
- Single Precision (32 bit): 1 sign bit, 8 bit exponent, and 23 bit mantissa
- Double Precision (64 bit): 1 sign bit, 11 bit exponent, and 52 bit mantissa
- Quadruple Precision (128 bit): 1 sign bit, 15 bit exponent, and 112 bit mantissa

Special Value Representation −

There are some special values depended upon different values of the exponent and mantissa in the IEEE 754 standard.

- All the exponent bits 0 with all mantissa bits 0 represents 0. If sign bit is 0, then +0, else -0.
- All the exponent bits 1 with all mantissa bits 0 represents infinity. If sign bit is 0, then +∞, else -∞.
- All the exponent bits 0 and mantissa bits non-zero represents denormalized number.
- All the exponent bits 1 and mantissa bits non-zero represents error.

3)Describe about Floating-Point Representation ?

This representation has fixed number of bits for integer part and for fractional part. For example, if given fixed-point representation is IIII.FFFF, then you can store minimum value is 0000.0001 and maximum value is 9999.9999. There are three parts of a fixed-point number representation: the sign field, integer field, and fractional field.

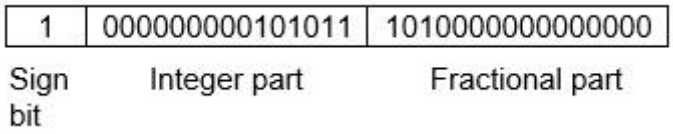
We can represent these numbers using:

- Signed representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 1's complement representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 2's complementation representation: range from $-(2^{(k-1)})$ to $(2^{(k-1)}-1)$, for k bits.

2's complementation representation is preferred in computer system because of unambiguous property and easier for arithmetic operations.

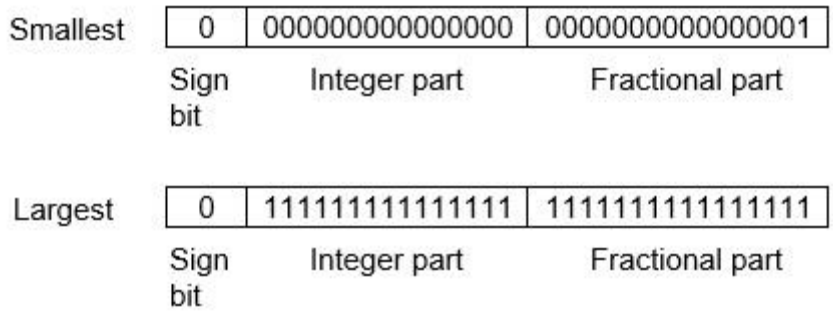
Example −Assume number is using 32-bit format which reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part.

Then, -43.625 is represented as following:



Where, 0 is used to represent + and 1 is used to represent. 000000000101011 is 15 bit binary value for decimal 43 and 1010000000000000 is 16 bit binary value for fractional 0.625.

The advantage of using a fixed-point representation is performance and disadvantage is relatively limited range of values that they can represent. So, it is usually inadequate for numerical analysis as it does not allow enough numbers and accuracy. A number whose representation exceeds 32 bits would have to be stored inexactly.



These are above smallest positive number and largest positive number which can be store in 32-bit representation as given above format. Therefore, the smallest positive number is $2^{-16} \approx 0.000015$ approximate and the largest positive number is $(2^{15}-1)+(1-2^{-16})=2^{15}(1-2^{-16}) =32768$, and gap between these numbers is 2^{-16}.

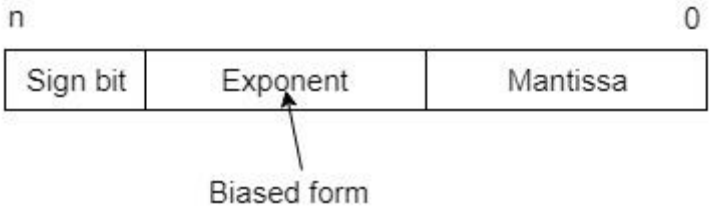
We can move the radix point either left or right with the help of only integer field is 1.

Floating-Point Representation −

This representation does not reserve a specific number of bits for the integer part or the fractional part. Instead it reserves a certain number of bits for the number (called the mantissa or significand) and a certain number of bits to say where within that number the decimal place sits (called the exponent).

The floating number representation of a number has two part: the first part represents a signed fixed point number called mantissa. The second part of designates the position of the decimal (or binary) point and is called the exponent. The fixed point mantissa may be fraction or an integer. Floating -point is always interpreted to represent a number in the following form: Mxr^e.

Only the mantissa m and the exponent e are physically represented in the register (including their sign). A floating-point binary number is represented in a similar manner except that is uses base 2 for the exponent. A floating-point number is said to be normalized if the most significant digit of the mantissa is 1.



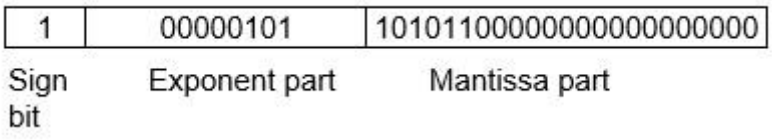
So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number.

Note that signed integers and exponent are represented by either sign representation, or one's complement representation, or two's complement representation.

The floating point representation is more flexible. Any non-zero number can be represented in the normalized form of $\pm(1.b_1b_2b_3\ ...)_2x2^n$ This is normalized form of a number x.

Example −Suppose number is using 32-bit format: the 1 bit sign bit, 8 bits for signed exponent, and 23 bits for the fractional part. The leading bit 1 is not stored (as it is always 1 for a normalized number) and is referred to as a *"hidden bit*".

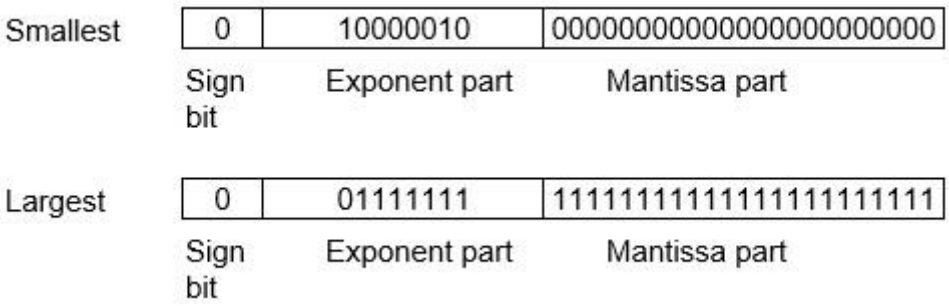
Then −53.5 is normalized as $-53.5=(-110101.1)_2=(-1.101011)x2^5$, which is represented as following below,



Where 00000101 is the 8-bit binary value of exponent value +5.

Note that 8-bit exponent field is used to store integer exponents $-126 \leq n \leq 127$.

The smallest normalized positive number that fits into 32 bits is $(1.00000000000000000000000)_2x2^{-126}=2^{-126}\approx1.18x10^{-38}$, and largest normalized positive number that fits into 32 bits is $(1.11111111111111111111111)_2x2^{127}=(2^{24}-1)x2^{104}\approx 3.40x10^{38}$. These numbers are represented as following below,



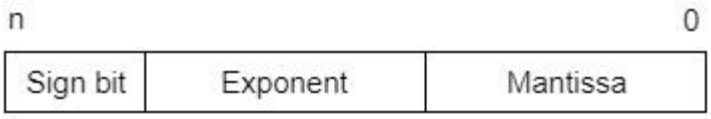
The precision of a floating-point format is the number of positions reserved for binary digits plus one (for the hidden bit). In the examples considered here the precision is 23+1=24.

The gap between 1 and the next normalized floating-point number is known as machine epsilon. the gap is $(1+2^{-23})-1=2^{-23}$for above example, but this is same as the smallest positive floating-point number because of non-uniform spacing unlike in the fixed-point scenario.

Note that non-terminating binary numbers can be represented in floating point representation, e.g., 1/3 = $(0.010101\ ...)_2$ cannot be a floating-point number as its binary representation is non-terminating.

IEEE Floating point Number Representation −

IEEE (Institute of Electrical and Electronics Engineers) has standardized Floating-Point Representation as following diagram.



So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number. The sign bit is 0 for positive number and 1 for negative number. Exponents are represented by or two's complement representation.

According to IEEE 754 standard, the floating-point number is represented in following ways:

- Half Precision (16 bit): 1 sign bit, 5 bit exponent, and 10 bit mantissa
- Single Precision (32 bit): 1 sign bit, 8 bit exponent, and 23 bit mantissa
- Double Precision (64 bit): 1 sign bit, 11 bit exponent, and 52 bit mantissa
- Quadruple Precision (128 bit): 1 sign bit, 15 bit exponent, and 112 bit mantissa

Special Value Representation −

There are some special values depended upon different values of the exponent and mantissa in the IEEE 754 standard.

- All the exponent bits 0 with all mantissa bits 0 represents 0. If sign bit is 0, then +0, else -0.
- All the exponent bits 1 with all mantissa bits 0 represents infinity. If sign bit is 0, then +∞, else -∞.
- All the exponent bits 0 and mantissa bits non-zero represents denormalized number.
- All the exponent bits 1 and mantissa bits non-zero represents error.

4)Describe about number system?

he technique to represent and work with numbers is called **number system**. **Decimal number system** is the most common number system. Other popular number systems include **binary number system, octal number system, hexadecimal number system,** etc.

Decimal Number System

Decimal number system is a **base 10** number system having 10 digits from 0 to 9. This means that any numerical quantity can be represented using these 10 digits. Decimal number system is also a **positional value system**. This means that the value of digits will depend on its position. Let us take an example to understand this.

Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−

- In 734, value of 7 is 7 hundreds or 700 or 7×100 or 7×10^2
- In 971, value of 7 is 7 tens or 70 or 7×10 or 7×10^1
- In 207, value 0f 7 is 7 units or 7 or 7×1 or 7×10^0

The weightage of each position can be represented as follows −

10^5	10^4	10^3	10^2	10^1	10^0

In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed. Let's look at them in detail.

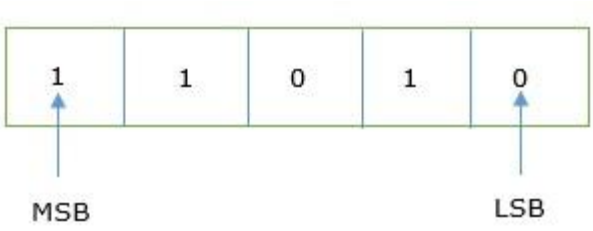
Binary Number System

The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two digits – 0 and 1 – is called **binary number system**.

Each binary digit is also called a **bit**. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here.

2^5	2^4	2^3	2^2	2^1	2^0

In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB)**.



And decimal equivalent of this number is sum of product of each digit with its positional value.

$11010_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 16 + 8 + 0 + 2 + 0$

$= 26_{10}$

Computer memory is measured in terms of how many bits it can store. Here is a chart for memory capacity conversion.

- 1 byte (B) = 8 bits
- 1 Kilobytes (KB) = 1024 bytes
- 1 Megabyte (MB) = 1024 KB
- 1 Gigabyte (GB) = 1024 MB
- 1 Terabyte (TB) = 1024 GB
- 1 Exabyte (EB) = 1024 PB
- 1 Zettabyte = 1024 EB
- 1 Yottabyte (YB) = 1024 ZB

Octal Number System

Octal number system has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here −

8^5	8^4	8^3	8^2	8^1	8^0

Decimal equivalent of any octal number is sum of product of each digit with its positional value.

$726_8 = 7\times8^2 + 2\times8^1 + 6\times8^0$

$= 448 + 16 + 6$

$= 470_{10}$

Hexadecimal Number System

Octal number system has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16, as shown here −

16^5	16^4	16^3	16^2	16^1	16^0

Decimal equivalent of any hexadecimal number is sum of product of each digit with its positional value.

$27FB_{16} = 2\times16^3 + 7\times16^2 + 15\times16^1 + 10\times16^0$

$= 8192 + 1792 + 240 +10$

$= 10234_{10}$

Number System Relationship

The following table depicts the relationship between decimal, binary, octal and hexadecimal number systems.

HEXADECIMAL	DECIMAL	OCTAL	BINARY
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110

7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

ASCII

Besides numerical data, computer must be able to handle alphabets, punctuation marks, mathematical operators, special symbols, etc. that form the complete character set of English language. The complete set of characters or symbols are called alphanumeric codes. The complete alphanumeric code typically includes −

- 26 upper case letters
- 26 lower case letters
- 10 digits
- 7 punctuation marks
- 20 to 40 special characters

Now a computer understands only numeric values, whatever the number system used. So all characters must have a numeric equivalent called the alphanumeric code. The most widely used alphanumeric code is American Standard Code for Information Interchange (ASCII). ASCII is a 7-bit code that has 128 (27) possible codes.

5)Explain about addition and subtractor algorithm?

Addition and Subtraction with Signed –Magnitude Data

We designate the magnitude of the two numbers by A and B. Where the signed numbers are added or subtracted, we find that there are eight different conditions to consider,

depending on the sign of the numbers and the operation performed. These conditions are listed in the first column of Table 4.1. The other columns in the table show the actual operation to be performed with the magnitude of the numbers. The last column is needed to present a negative zero. In other words, when two equal numbers are subtracted, the result should be +0 not -0.

The algorithms for addition and subtraction are derived from the table and can be stated as follows (the words parentheses should be used for the subtraction algorithm)

SIGNED MAGNITUDEADDITION AND SUBTRACTION

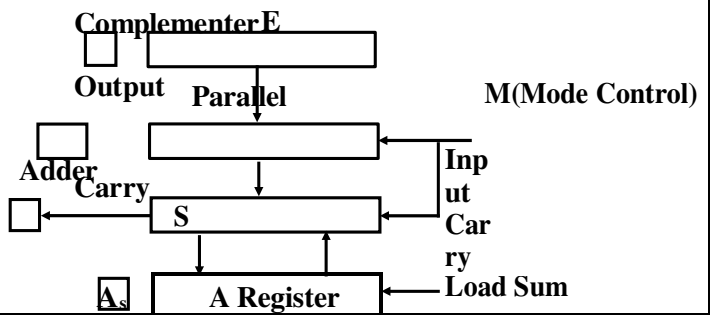
Addition: A + B ; A: Augend; B: Addend

Subtraction: A - B: A: Minuend; B: Subtrahend

Operation	Add Magnitude	Subtract Magnitude When A>B	When A<B	When A=B
(+A) + (+B)	+(A + B)			
(+A) + (- B)		+(A - B)	- (B - A)	+(A - B)
(- A) + (+B)		- (A - B)	+(B - A)	+(A - B)
(- A) + (- B)	- (A + B)			
(+A) - (+B)		+(A - B)	- (B - A)	+(A - B)
(+A) - (- B)	+(A + B)			
(- A) - (+B)	- (A + B)			
(- A) - (- B)		- (A - B)	+(B - A)	+(A - B)

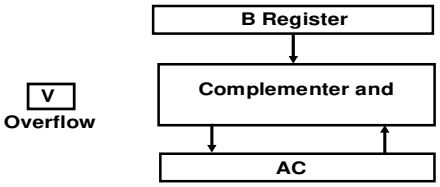
Hardware Implementation B_S B Register

AVF

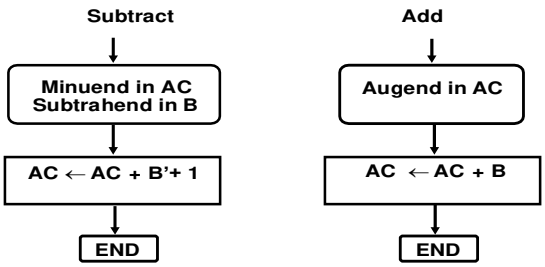


SIGNED 2'S COMPLEMENT ADDITION AND SUBTRACTION

Hardware



Algorithm



Algorithm:

- The flowchart is shown in Figure 7.1. The two signs A, and B, are compared by anexclusive-OR gate.

  If the output of the gate is 0 the signs are identical; If it is 1, the signs are different.

- For an add operation, identical signs dictate that the magnitudes be added. For a subtract operation, different signs dictate that the magnitudes be added.

- The magnitudes are added with a microoperation EA A + B, where EA is a register that combines E and A. The carry in E after the addition constitutes an overflow if it is equal to 1. The value of E is transferred into the add-overflow flip-flop AVF.

- The two magnitudes are subtracted if the signs are different for an add operation or identical for a subtract operation. The magnitudes are subtracted by adding A to the 2's complemented B. No overflow can occur if the numbers are subtracted so AVF is cleared to 0.

- 1 in E indicates that A >= B and the number in A is the correct result. If this numbs is zero, the sign A must be made positive to avoid a negative zero.

- 0 in E indicates that A < B. For this case it is necessary to take the 2's complement of the value in A. The operation can be done with one microoperation A A' +1.

- However, we assume that the A register has circuits for microoperations complement and increment, so the 2's complement is obtained from these two microoperations.

- In other paths of the flowchart, the sign of the result is the same as the sign of A. so no change in A is required. However, when A < B, the sign of the result is the complement of the original sign of A. It is then necessary to complement A, to obtain the correct sign.

- The final result is found in register A and its sign in As. The value in AVF provides an overflow indication. The final value of E is immaterial.

- a block diagram of the hardware for implementing the addition andsubtraction operations.

  - It consists of registers A and B and sign flip-flops As and Bs.
  - Subtraction is done by adding A to the 2's complement of B.

- The output carry is transferred to flip-flop E , where it can be checked to determine the relative magnitudes of two numbers.

- The add-overflow flip-flop *AVF* holds the overflow bit when A and B are added.

- The A register provides other microoperations that may be needed when we specify the sequence of steps in the algorithm.

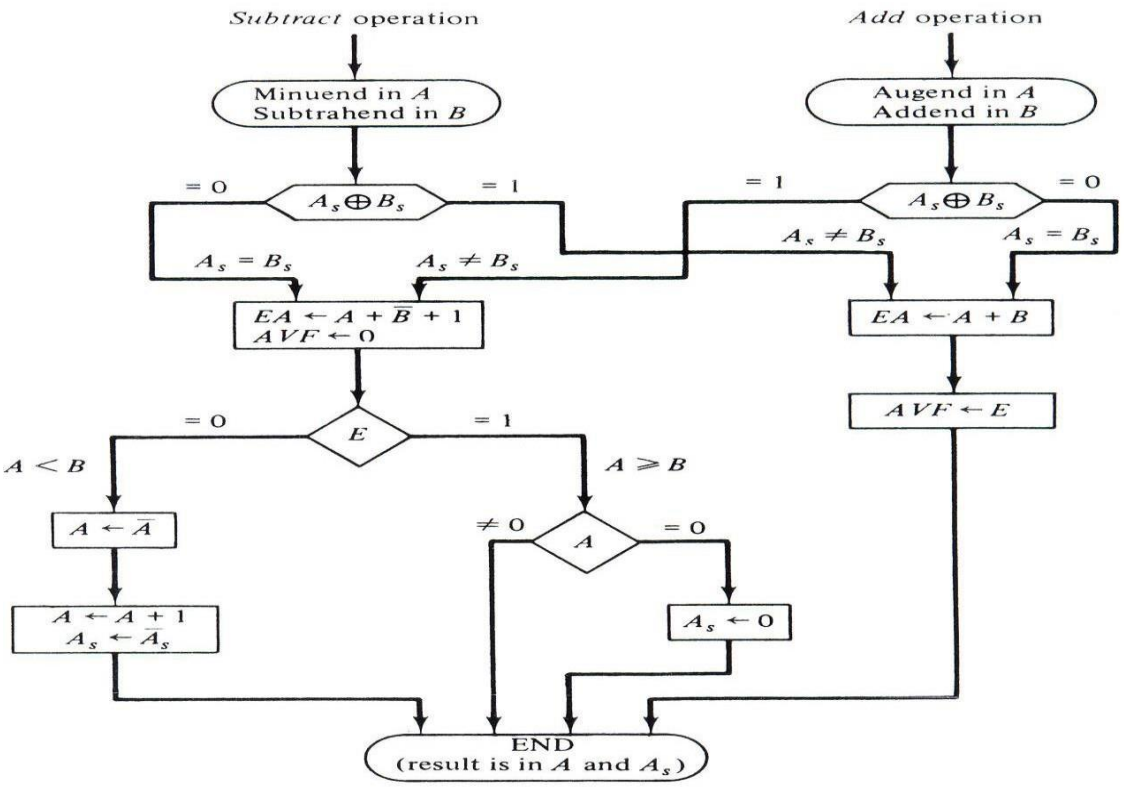
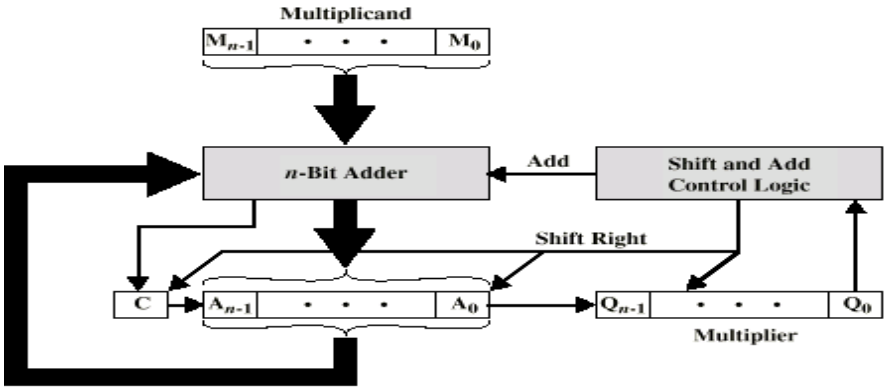


Figure 10-2 Flowchart for add and subtract operations.

6) Explain about Multiplication Algorithm?

In the beginning, the multiplicand is in B and the multiplier in Q. Their corresponding signs are in Bs and Qs respectively. We compare the signs of both A and Q and set to corresponding sign of the product since a double-length product will be stored in registers A and Q. Registers A and E are cleared and the sequence counter SC is set to the number of bits of the multiplier. Since an operand must be stored with its sign, one bit of the word will be occupied by the sign and the magnitude will consist of n-1 bits.

Now, the low order bit of the multiplier in Qn is tested. If it is 1, the multiplicand (B) is added to present partial product (A), 0 otherwise. Register EAQ is then shifted once to the right to form the new partial product. The sequence counter is decremented by 1 and its new value checked. If it is not equal to zero, the process is repeated and a new partial product is formed. When SC = 0 we stops the process.



C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

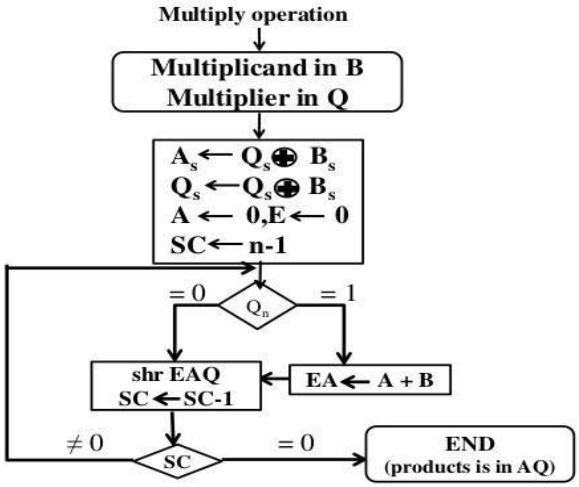


Figure: Flowchart for multiply operation.

Booth's algorithm :

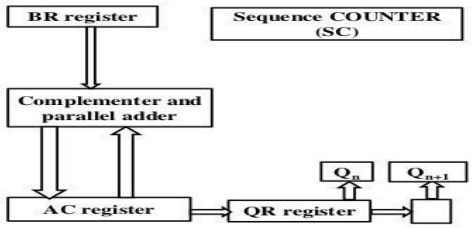
- Booth algorithm gives a procedure for multiplying binary integers in signed- 2's complement representation.
- It operates on the fact that strings of 0's in the multiplier require no addition but just

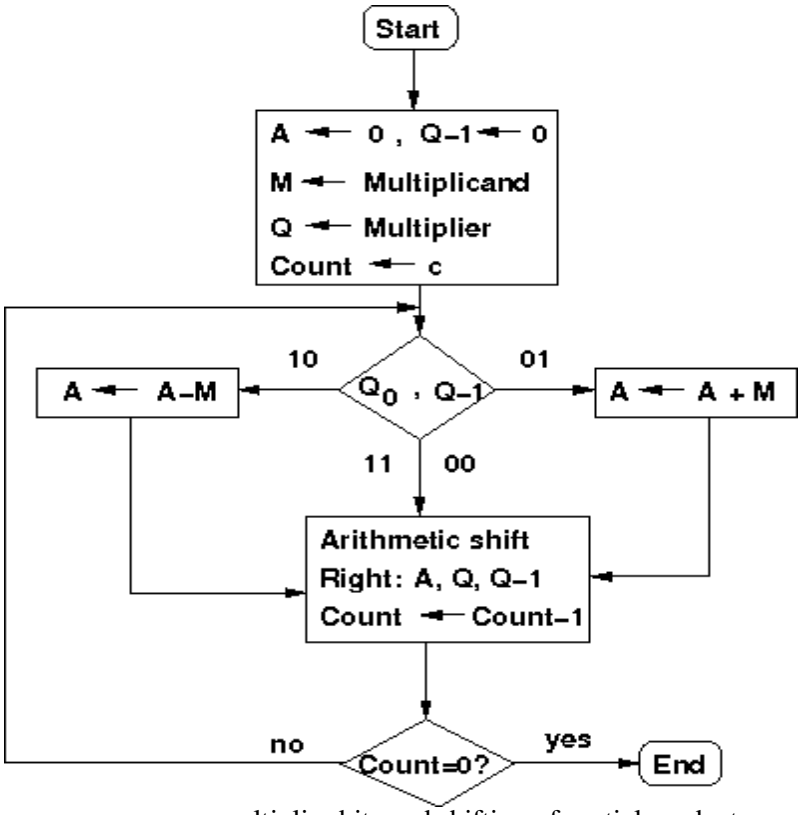
shifting, and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$.

- For example, the binary number 001110 (+14) has a string 1's from 2^3 to 2^1 (k=3, m=1). The number can be represented as $2^{k+1} - 2^m$. = $2^4 - 2^1$ = 16 – 2 = 14. Therefore, the multiplication M X 14, where M is the multiplicand and 14 the multiplier, can be done as M X 2^4 – M X 2^1.
- Thus the product can be obtained by shifting the binary multiplicand M four times to the left and subtracting M shifted left once.
- As in all multiplication schemes, booth algorithm requires examination of the

Hardware for Booth Algorithm

- Sign bits are not separated from the rest of the registers
- rename registers A,B, and Q as AC,BR and QR respectively
- Q_n designates the least significant bit of the multiplier in register QR
- Flip-flop Qn+1 is appended to QR to facilitate a double bit inspection of the multiplier





multiplier bits and shifting of partial product.

- Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial, or left unchanged according to the following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.

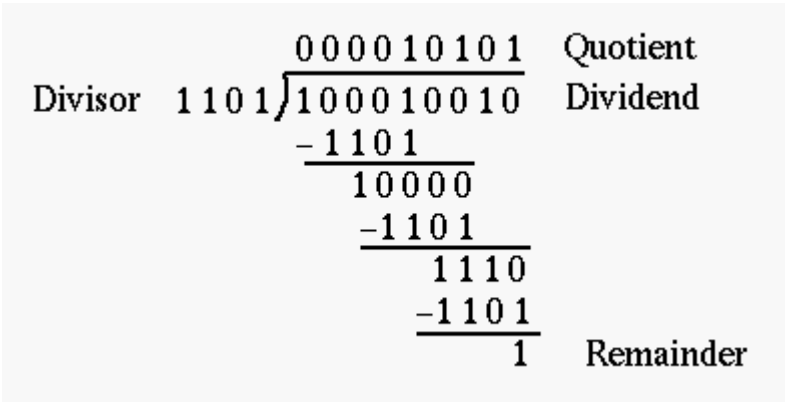
2. The multiplicand is added to the partial product upon encountering the first 0 in a string of 0's in the multiplier.

3. The partial product does not change when multiplier bit is identical to the previous multiplier bit.

- The algorithm works for positive or negative multipliers in 2's complement representation.
- This is because a negative multiplier ends with a string of 1's and the last operation will be a subtraction of the appropriate weight.
- The two bits of the multiplier in Qn and Qn+1 are inspected.
- If the two bits are equal to 10, it means that the first 1 in a string of 1 's has been encountered. This requires a subtraction of the multiplicand from the partial product in AC.
- If the two bits are equal to 01, it means that the first 0 in a string of 0's has been encountered. This requires the addition of the multiplicand to the partial product in AC.
- When the two bits are equal, the partial product does not change.

8)Expalin about Division Algorithms?

Division of two fixed-point binary numbers in signed magnitude representation is performed with paper and pencil by a process of successive compare, shift and subtract operations. Binary division is much simpler than decimal division because here the quotient digits are either 0 or 1 and there is no need to estimate how many times the dividend or partial remainder fits into the divisor. The division process is described in Figure



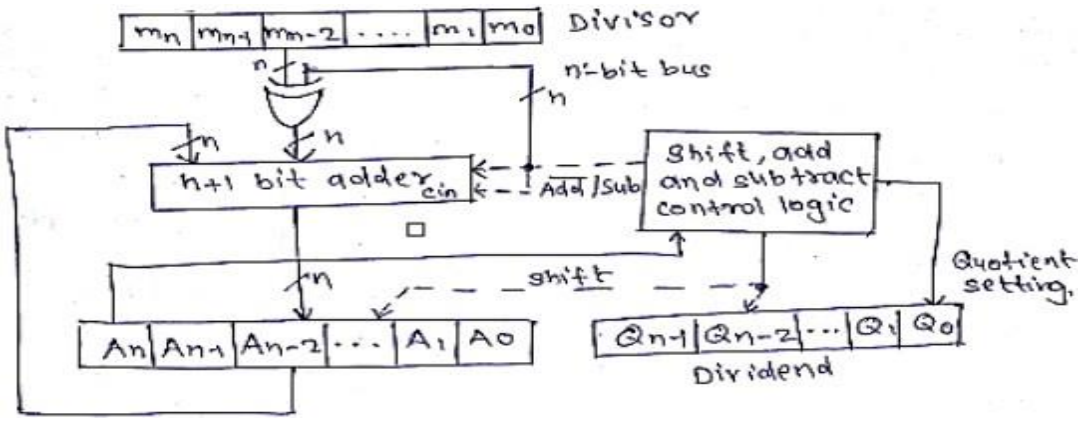
The devisor is compared with the five most significant bits of the dividend. Since the 5-bit number is smaller than B, we again repeat the same process. Now the 6-bit number is greater than B, so we place a 1 for the quotient bit in the sixth position above the dividend. Now we shift the divisor once to the right and subtract it from the dividend. The difference is known as a partial remainder because the division could have stopped here to obtain a quotient of 1 and a remainder equal to the partial

remainder. Comparing a partial remainder with the divisor continues the process. If the partial remainder is greater than or equal to the divisor, the quotient bit is equal to 1. The divisor is then shifted right and subtracted from the partial remainder. If the partial remainder is smaller than the divisor, the quotient bit is 0 and no subtraction is needed. The divisor is shifted once to the right in any case. Obviously the result gives both a quotient and a remainder.

Hardware Implementation for Signed-Magnitude Data

In hardware implementation for signed-magnitude data in a digital computer, it is convenient to change the process slightly. Instead of shifting the divisor to the right, two dividends, or partial remainders, are shifted to the left, thus leaving the two numbers in the required relative position. Subtraction is achieved by adding A to the 2's complement of B. End carry gives the information about the relative magnitudes.

The hardware required is identical to that of multiplication. Register EAQ is now shifted to the left with 0 inserted into Qn and the previous value of E is lost. The example is given in Figure 4.10 to clear the proposed division process. The divisor is stored in the B register and the double-length dividend is stored in registers A and Q. The dividend is shifted to the left and the divisor is subtracted by adding its 2's complement value. E



Hardware Implementation for Signed-Magnitude Data

7)Describe about Addition and Subtraction of Floating Point Numbers?

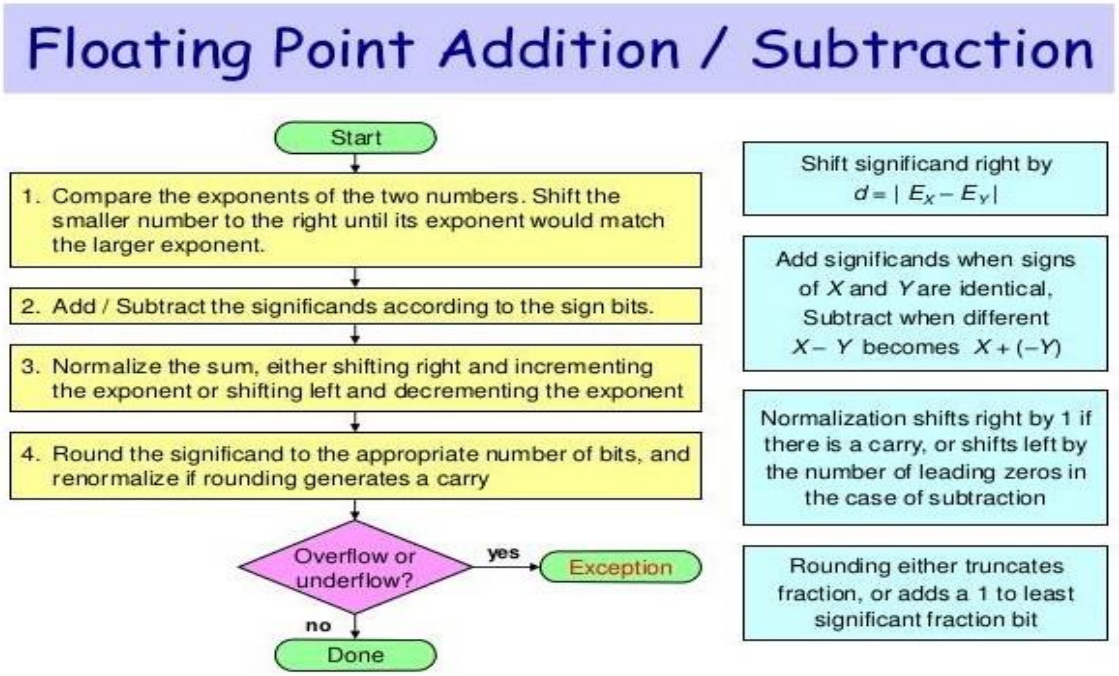
During addition or subtraction, the two floating-point operands are kept in AC and BR. The sum or difference is formed in the AC. The algorithm can be divided into four consecutive parts:

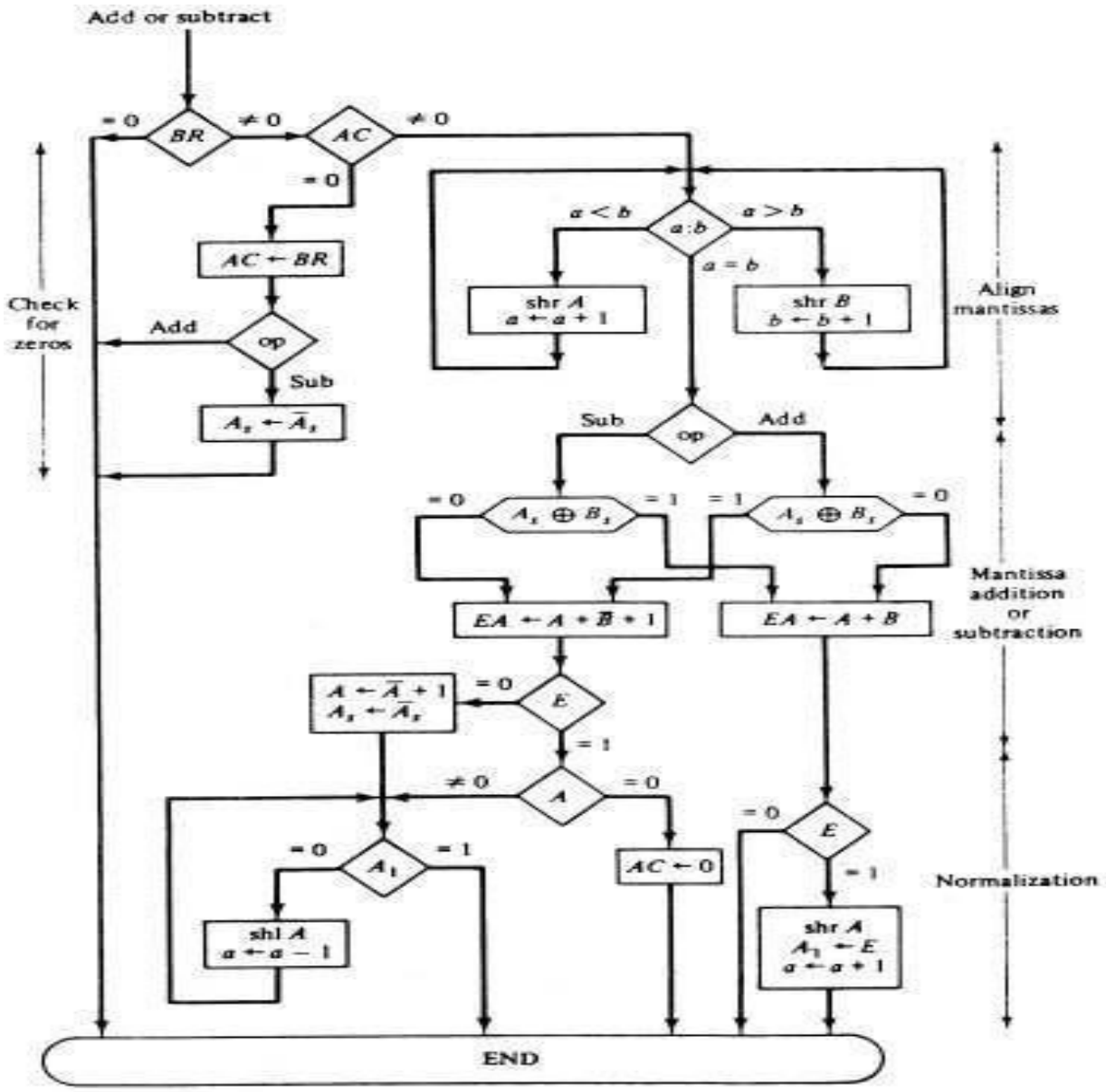
1. Check for zeros.
2. Align the mantissas.
3. Add or subtract the mantissas

4. Normalize the result

A floating-point number cannot be normalized, if it is 0. If this number is used for computation, the result may also be zero. Instead of checking for zeros during the normalization process we check for zeros at the beginning and terminate the process if necessary. The alignment of the mantissas must be carried out prior to their operation. After the mantissas are added or subtracted, the result may be un-normalized. The normalization procedure ensures that the result is normalized before it is transferred to memory.

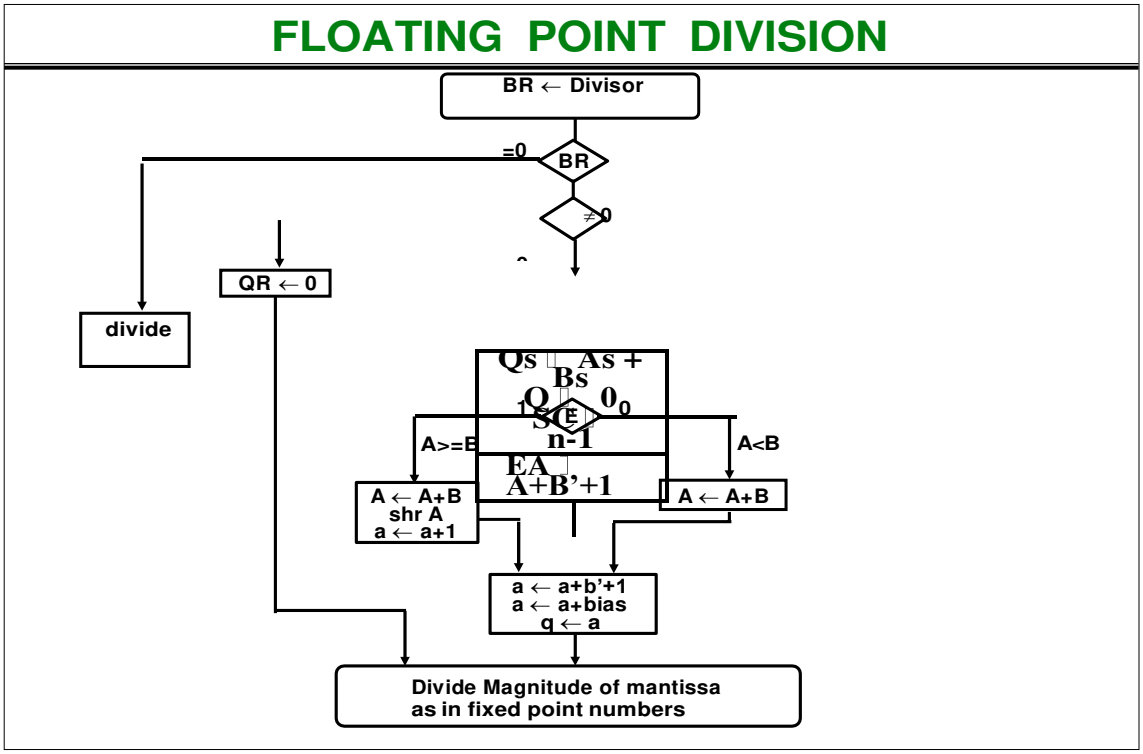
If the magnitudes were subtracted, there may be zero or may have an underflow in the result. If the mantissa is equal to zero the entire floating-point number in the AC is cleared to zero. Otherwise, the mantissa must have at least one bit that is equal to 1. The mantissa has an underflow if the most significant bit in position A1, is 0. In that case, the mantissa is shifted left and the exponent decremented. The bit in A1 is checked again and the process is repeated until A1 = 1. When A1 = 1, the mantissa is normalized and the operation is completed.





Algorithm for Floating Point Addition and Subtraction

Multiplication



UNIT IV

Short answers:

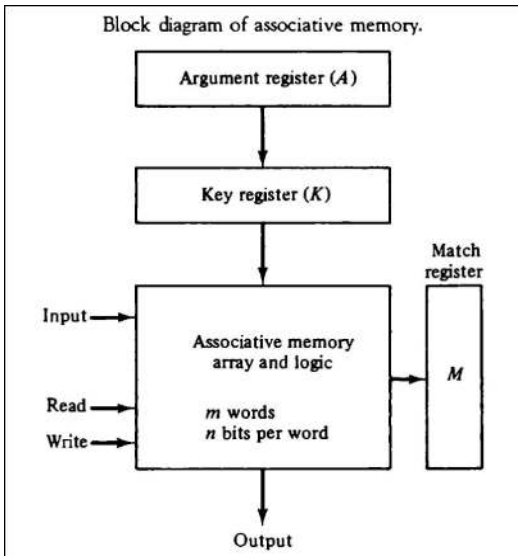
1)Define auxiliary memory?

The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called **auxiliary memory**.

2)Define bootstrap loader?

The ROM portion of main memory is needed for storing an initial program called a **bootstrap loader**. The bootstrap loader is a program whose function is to start the computer software operating when power is turned on.

3)Draw a block diagram for associative memory?



4)Define tag register?

If unwanted words have to be deleted and new words inserted one at a time, there is a need for aspecial register to distinguish between active and inactive words. This register, sometimes called a *tag register.*

5)Define hit ratio?

The performance of cache memory is frequently measured in terms of a quantity called **hit ratio**. When the CPU refers to memory and finds the word in cache, it is said to produce a hit. If the word is not found in cache, it is in main memory and it counts as a miss. The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is thehit ratio.

6)What are the three mapping procedures?

Three types of mapping procedures are :

1. Associative mapping
2. Direct mapping
3. Set-associative mapping.

7)Define memory mapped i/o?

The other alternative is to use the same address space for both memory and I/O. This is the case in computers that employ only one set of read and write signals and do not distinguish between memory and I/O addresses. This configuration is referred to as **memory mapped I/O**. The computer treats an interface register as being part of the memory system.

8)Define handshaking?

The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as **handshaking.**

9)Write the three possible modes in data transfer?

Data transfer to and from peripherals may be handled in one of three possible modes:

1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct memory access (DMA)

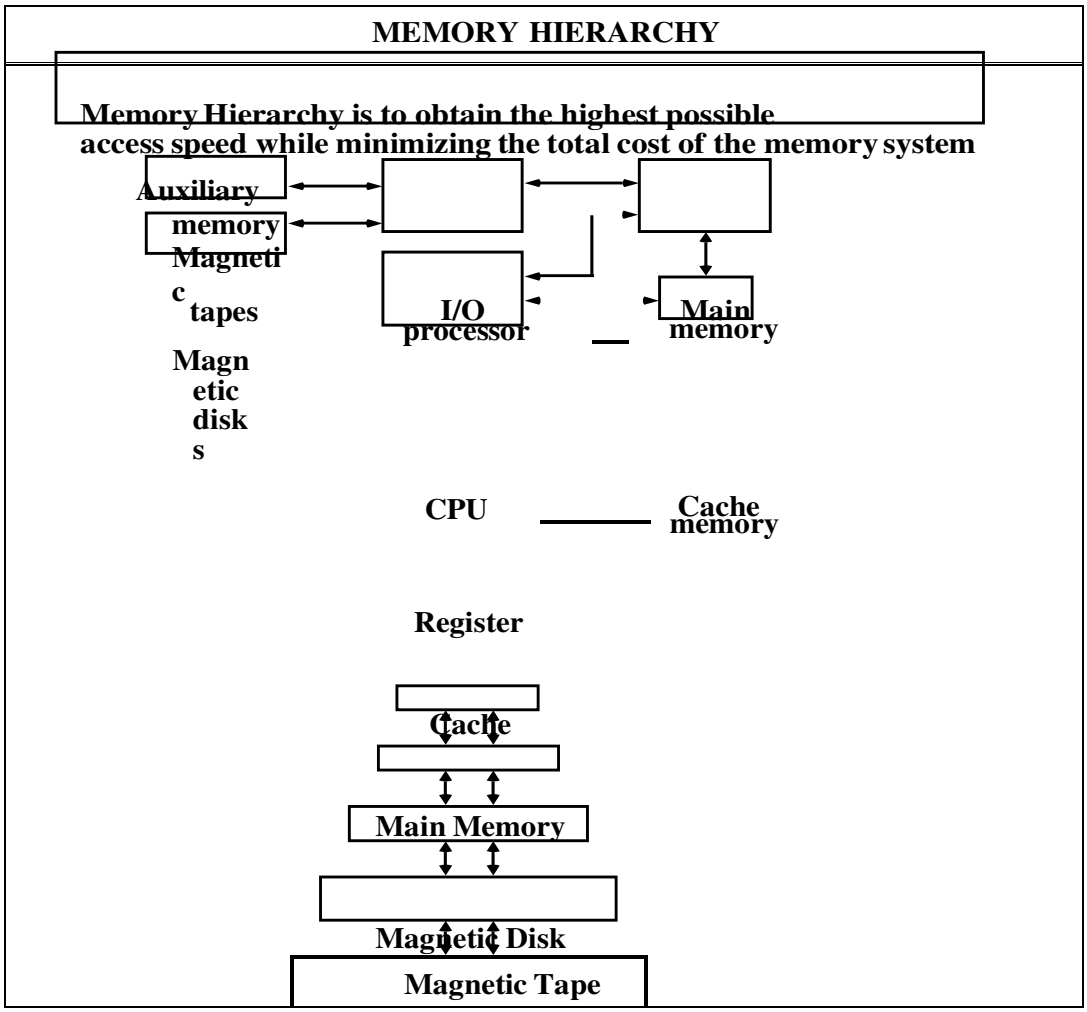
10)Define DMA?

The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called **direct memory access (DMA)**.

LONG ANSWERS:

1)Expalin about memory hierarchy?

Memory Hierarchy :



Main Memory

- The main memory is the central storage unit in a computer system.
- Primary memory holds only those data and instructions on which computer is currently working.

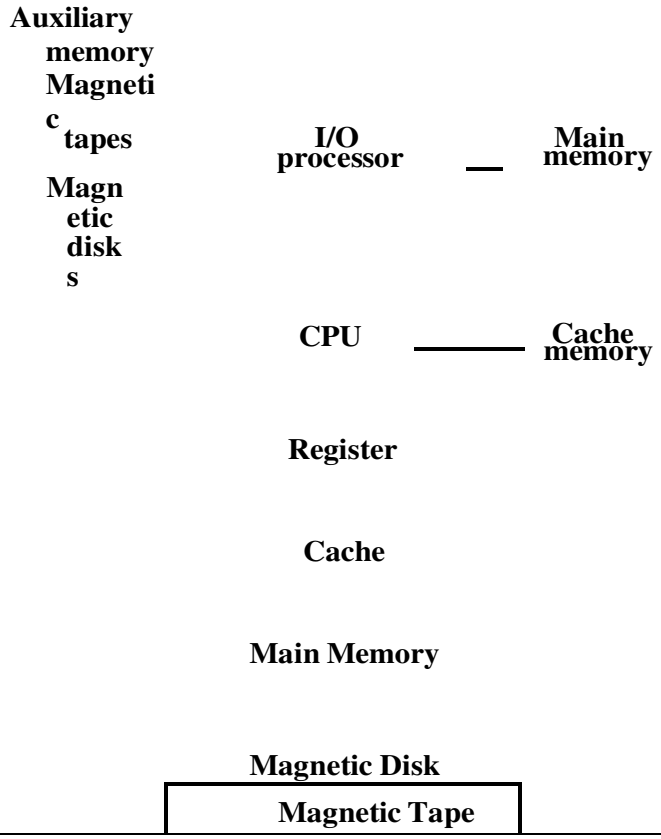
- It has limited capacity and data is lost when power is switched off.
- It is generally made up of semiconductor device.
- These memories are not as fast as registers.
- The data and instruction required to be processed reside in main memory.
- It is divided into two subcategories RAM and ROM.

Memory address map of RAM and ROM

- The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM.
- The interconnection between memory and processor is then established from knowledge of the size of memory needed and the type of RAM and ROM chips available.
- The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip.
- The table, called a **memory address map**, is a pictorial representation of assigned address space for each chip in the system,

MEMORY HIERARCHY

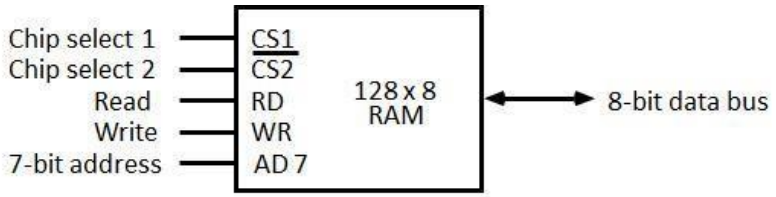
Memory Hierarchy is to obtain the highest possible access speed while minimizing the total cost of the memory system

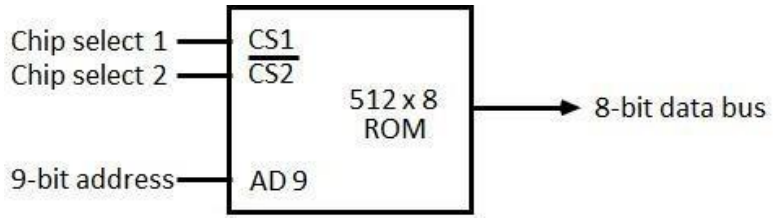


 To demonstrate with a particular example, assume that a computer system needs 512 bytes ofRAM and 512 bytes of ROM.

 The RAM and ROM chips to be used are specified in figure 9.1 and figure 9.2.

Memory address map of RAM and ROM





Component	Hexa address	Address bus 10 9 8 7 6 5 4 3 2 1
RAM 1	0000 - 007F	0 0 0 x x x x x x x
RAM 2	0080 - 00FF	0 0 1 x x x x x x x
RAM 3	0100 - 017F	0 1 0 x x x x x x x
RAM 4	0180 - 01FF	0 1 1 x x x x x x x
ROM	0200 - 03FF	1 x x x x x x x x x

 The component column specifies whether a RAM or a ROM chip is used.

 The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip.

 The address bus lines are listed in the third column.

 Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero.

 The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip.

 The RAM chips have 128 bytes and need seven address lines. The ROM chip has 512 bytes and needs 9 address lines.

 The x's are always assigned to the low-order bus lines: lines 1 through 7 for the RAM and lines 1 through 9 for the ROM.

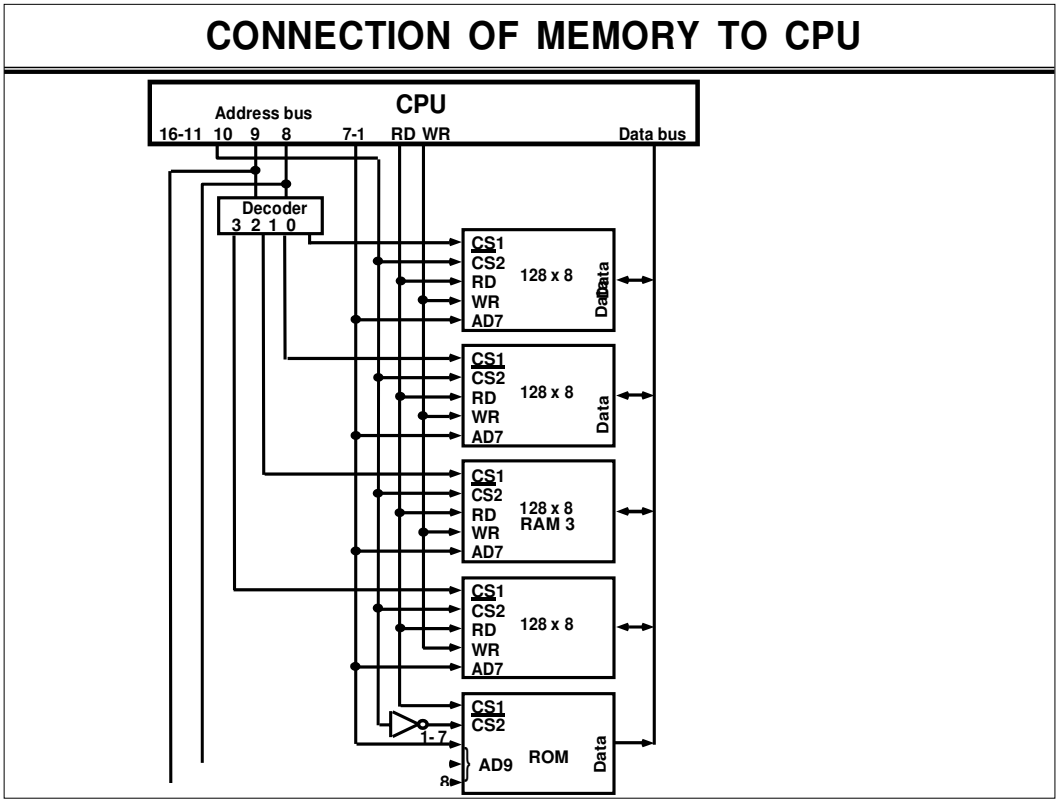
- It is now necessary to distinguish between four RAM chips by assigning to each a different address. For this particular example we choose bus lines 8 and 9 to represent four distinct binary combinations.
- The table clearly shows that the nine low-order bus lines constitute a memory space for RAM equal to $2^9 = 512$ bytes.
- The distinction between a RAM and ROM address is done with another bus line. Here we choose line 10 for this purpose.

When line 10 is 0, the CPU selects a RAM, and when this line is equal to 1, it selects the ROM

Memory connections to CPU :

- RAM and ROM chips are connected to a CPU through the data and address buses

- The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs.



2) Give a brief explain about auxiliary memory?

- **Magnetic Tape:** Magnetic tapes are used for large computers like mainframe computers where large volume of data is stored for a longer time. In PC also you can use tapes in the form of cassettes. The cost of storing data in tapes is inexpensive. Tapes consist of magnetic materials that store data permanently. It can be 12.5 mm to 25 mm wide plastic film-type and 500 meter to 1200 meter long which is coated with magnetic material. The deck is connected to the central processor and information is fed into or read from the tape through the processor. It's similar to cassette tape recorder

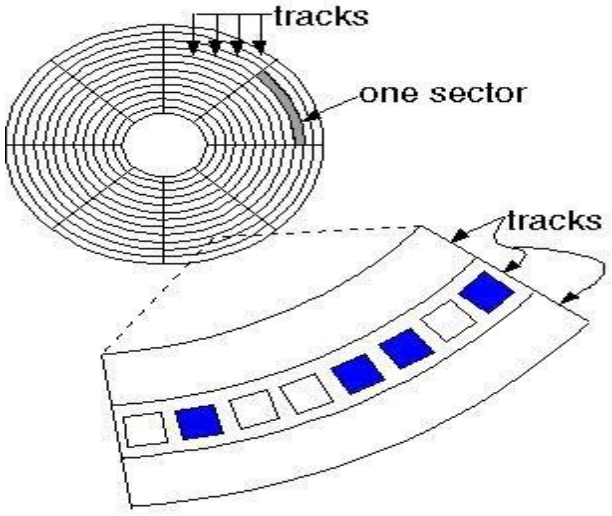
Magnetic tape is an information storage medium consisting of a magnetisable coating on a thin plastic strip. Nearly all recording tape is of this type, whether used for video with a video cassette recorder, audio storage (reel-to-reel tape, compact audio cassette, digital audio tape (DAT), digital linear tape (DLT) and other formats including 8-track cartridges) or general purpose digital data storage using a computer (specialized tape formats, as well as the above-mentioned compact audio cassette, used with home computers of the 1980s, and DAT, used for backup in workstation installations of the 1990s).

- Magneto-optical and optical tape storage products have been developed using many of the same concepts as magnetic storage, but have achieved little commercial success.

- **Magnetic Disk:** You might have seen the gramophone record, which is circular like a disk and coated with magnetic material. Magnetic disks used in computer are made on the same principle. It rotates with very high speed inside the computer drive. Data is stored on both the surface of the disk. Magnetic disks are most popular for direct access storage device. Each disk consists of a number of invisible concentric circles called tracks. Information is recorded on tracks of a disk surface in the form of tiny magnetic spots. The presence of a magnetic spot represents one bit and its absence represents zero bit. The information stored in a disk can be read many times without affecting the stored data. So the reading operation is non-destructive. But if you want to write a new data, then the existing data is erased from the disk and new data is recorded. For Example-Floppy Disk.

Sectors

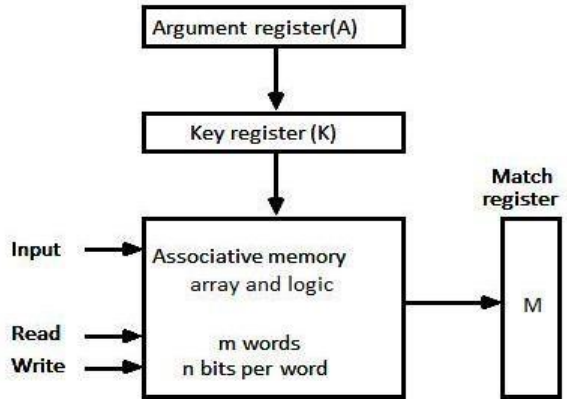
Tracks are further divided into sectors, which hold a block of data that is read or written at one time; for example, READ SECTOR 782, WRITE SECTOR 5448. In order to update the disk, one or more sectors are read into the computer, changed and written back to disk. The operating system figures out how to fit data into these fixed spaces. Modern disks have more sectors in the outer tracks than the inner ones because the outer radius of the platter is greater than the inner radius



Optical Disk: With every new application and software there is greater demand for memory capacity. It is the necessity to store large volume of data that has led to the development of optical disk storage medium. Optical disks can be divided into the following categories:

Associative Memory :Content Addressable Memory (CAM).

- The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.
- A memory unit accessed by content is called an associative memory or content addressable memory (CAM).
- This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.
- TheIt consists of a memory array and logic form words with n bits per word.
- The argument register A and key register K each have n bits, one for each bit of a word.
- The match register M has m bits, one for each memory word.



Each word in memory is compared in parallel with the content of the argument register.

- The words that match the bits of the argument register set a corresponding bit in the match register.
- After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.
- Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.

Hardware Organization

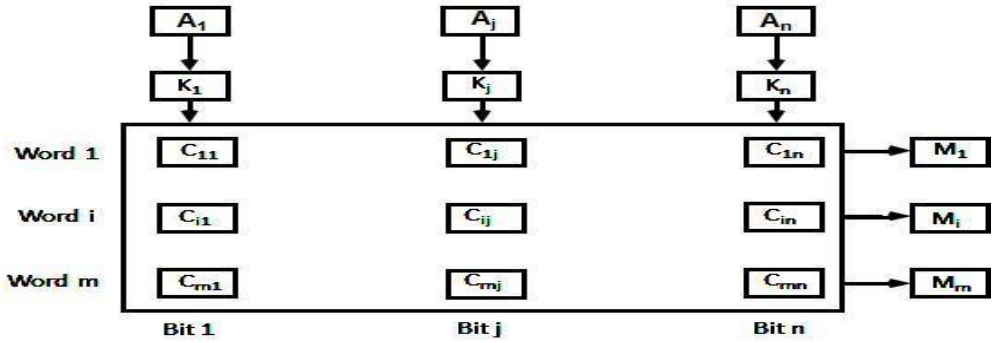
- The key register provides a mask for choosing a particular field or key in the argument word.
- The entire argument is compared with each memory word if the key register contains all 1's.

Otherwise, only those bits in the argument that have 1st in their corresponding position of the keyregister are compared.

- Thus the key provides a mask or identifying piece of information which specifies how thereference to memory is made.
- To illustrate with a numerical example, suppose that the argument register A and the key registerK have the bit configuration shown below.
- Only the three leftmost bits of A are compared with memory words because K has 1's in theseposition.

A	101 111100	
K	111 000000	
Word1	100 111100	no match
Word2	101 000001	match

- Word 2 matches the unmasked argument field because the three leftmost bits of the argumentand the word are equal.



- The relation between the memory array and external registers in an associative memory is shownin figure 9.4.
- The cells in the array are marked by the letter C with two subscripts.
- The first subscript gives the word number and the second specifies the bit position in

the word. - Thus cell Cij is the cell for bit j in words i.

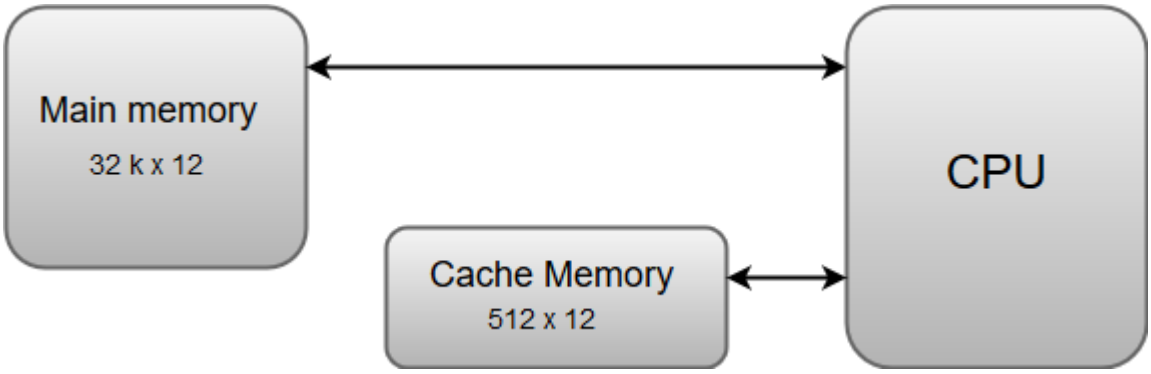
- A bit Aj in the argument register is compared with all the bits in column j of the array providedthat Kj =1.
- This is done for all columns j = 1, 2... n.
- If a match occurs between all the unmasked bits of the argument and the bits in word i, thecorresponding bit Mi in the match register is set to 1.

- If one or more unmasked bits of the argument and the word do not match, Mi is cleared to 0.

3) Explain about cache memory?

The data or contents of the main memory that are used frequently by CPU are stored in the cache memory so that the processor can easily access that data in a shorter time. Whenever the CPU needs to access memory, it first checks the cache memory. If the data is not found in cache memory, then the CPU moves into the main memory.

Cache memory is placed between the CPU and the main memory. The block diagram for a cache memory can be represented as:



The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

The basic operation of a cache memory is as follows:

- When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory.
- If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.
- A block of words one just accessed is then transferred from main memory to cache memory. The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed.
- The performance of the cache memory is frequently measured in terms of a quantity called **hit ratio**.
- When the CPU refers to memory and finds the word in cache, it is said to produce a **hit**.
- If the word is not found in the cache, it is in main memory and it counts as a **miss**.
- The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio.

4)Describe about modes of transfer?

- **Binary information** received from an external device is usually stored in memory for later processing. Information transferred from the central computer into an external device originates in the memory unit.
- **The CPU merely** executes the I/O instructions and may accept the data temporarily, but the ultimate source or destination is the memory unit.
- **Data transfer** between the central computer and I/O devices may be handled in a variety of modes.
- **Some modes** use the CPU as an intermediate path; others transfer the data directly to and from the memory unit.
- **Data transfer** to and from peripherals may be handled in one of three possible modes:

- 1. Programmed I/O
-
- 2. Interrupt-initiated I/O
-
- 3. Direct memory access (DMA)

-

- **Programmed I/O** operations are the result of I/O instructions written in the computer program.
- **Each data item** transfer is initiated by an instruction in the program.
- **Usually, the transfer** is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory. Transferring data under program control requires constant monitoring of the peripheral by the CPU.
- **Once a data transfer** is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. It is up to the programmed instructions executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device.

- **In the programmed** I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly.
- **It can be avoided** by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device. In the meantime the CPU can proceed to execute another program.

- **The interface** meanwhile keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer. Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing.
- **Transfer of data under** programmed I/O is between CPU and peripheral.
- **In direct memory access (DMA)**, the interface transfers data into and out of the memory unit through the memory bus. The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks.
- **When the transfer is made**, the DMA requests memory cycles through the memory bus.
- **When the request** is granted by the memory controller, the DMA transfers the data directly into memory. The CPU merely delays its memory access operation to allow the direct memory I/O transfer.
- **Since peripheral speed** is usually slower than processor speed, I/O-memory transfers are infrequent compared to processor access to memory.

5) Describe about Example of Programmed I/O?

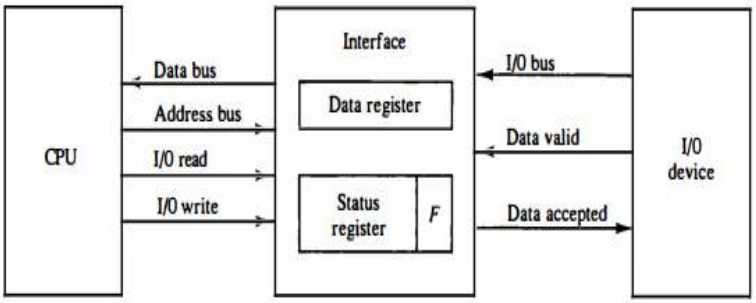
- **In the programmed I/O method**, the I/O device does not have direct access to memory.
- **A transfer from an I/O** device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from the device to the CPU and a store instruction to transfer the data from the CPU to memory.
- **Other instructions** may be needed to verify that the data are available from the device and to count the numbers of words transferred.
- **An example of data transfer** from an I/O device through an interface into the CPU . The device transfers bytes of data one at a time as they are available.
- **When a byte of data** is available, the device places it in the I/O bus and enables its data valid line.
- **The interface accepts** the byte into its data register and enables the data accepted line.

- **The interface sets** a bit in the status register that we will refer to as an F or "flag" bit. The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled by the interface.
- **This is according** to the handshaking procedure established in Fig. 5.
- **A program is written** for the computer to check the flag in the status register to determine if a byte has been placed in the data register by the I/O device.
- **This is done by reading** the status register into a CPU register and checking the value of the flag bit.
- **If the flag is equal** to 1, the CPU reads the data from the data reg
- **The flag bit** is then cleared to 0 by either the CPU or the interface, depending on how the interface circuits are designed.
- **Once the flag is cleared**, the interface disables the data accepted line and the device can then transfer the next data byte.
- **A flowchart of the program** that must be written for the CPU is shown in Fig. 11. It is assumed that the device is sending a sequence of bytes that must be stored in memory.
- **The transfer of each byte requires three instructions:**

- 1. Read the status register.
- 2. Check the status of the flag bit and branch to step 1 if not set or to step
- 3 if set.
- 3. Read the data register.

- **Each byte** is read into a CPU register and then transferred to memory with a store instruction. A common I/O programming task is to transfer a block of words from an I/O device and store them in a memory buffer.

Figure 10 Data transfer from I/O device to CPU.



F = Flag bit

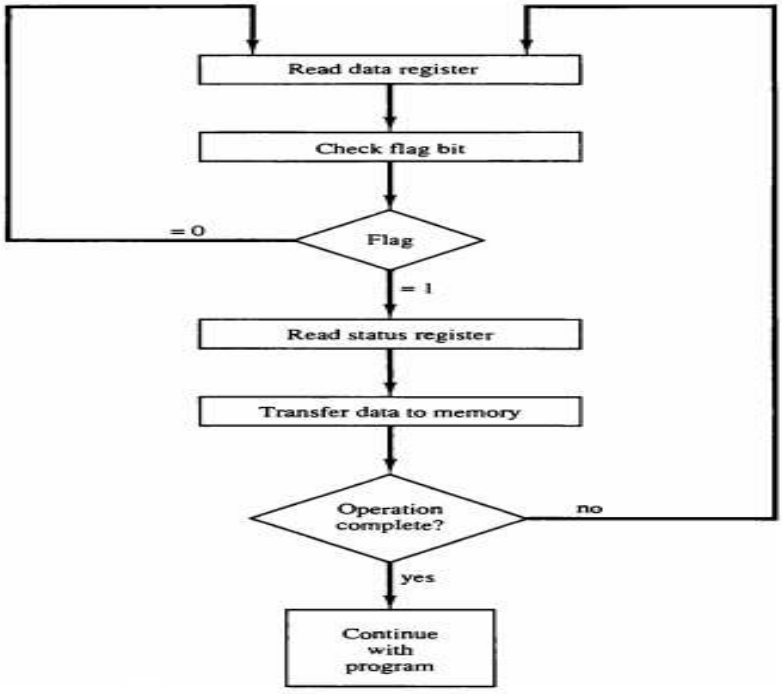


Figure 11 Flowchart for CPU program to input data.

- **The programmed** I/O method is particularly useful in small low-speed computers or in systems that are dedicated to monitor a device continuously.
- **The difference** in information transfer rate between the CPU and the I/O device makes this type of transfer inefficient.
- **To see why this** is inefficient, consider a typical computer that can execute the two instructions that read the status register and check the flag in 1 μS.
- **Assume that** the input device transfers its data at an average rate of 100 bytes per second.
- **This is equivalent to one byte** every 10,000 μS.
- **This means** that the CPU will check the flag 10,000 times between each transfer.
- **The CPU is wasting** time while checking the flag instead of doing some other useful processing task.

6) Describe about Interrupt-Initiated I/O?

- **An alternative** to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data.
- **This mode of transfer** uses the interrupt facility. While the CPU is running a program, it does not check the flag.
- **However**, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
- **The CPU deviates** from what it is doing to take care of the input or output transfer.
- **After the transfer** is completed, the computer returns to the previous program to continue what it was doing before the interrupt.
- **The CPU responds** to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.
- **The way that the processor chooses** the branch address of the service routine varies from one unit to another.
- **In principle**, there are two methods for accomplishing this.
- **One is called vectored** interrupt and the other, no vectored interrupt. In a non vectored interrupt, the branch address is assigned to a fixed location in memory.
- **In a vectored interrupt**, the source that interrupts supplies the branch information to the computer. This information is called the interrupt vector.
- **In some computers the interrupt vector** is the first address of the I/O service routine.
- **In other computers the interrupt vector** is an address that points to a location in memory where the beginning address of the I/O service routine is stored.

7)Decribe about Priority Interrupt?

- **Data transfer** between the CPU and an I/O device is initiated by the CPU. However, the CPU cannot start the transfer unless the device is ready to communicate with the CPU.

- **The readiness** of the device can be determined from an interrupt signal. The CPU responds to the interrupt request by storing the return address from PC into a memory stack and then the program branches to a service routine that processes the required transfer.
- **Some processors** also push the current PSW (program status word) onto the stack and load a new PSW for the service routine.
- **In a typical application** a number of I/O devices are attached to the computer, with each device being able to originate an interrupt request. The first task of the interrupt system is to identify the source of the interrupt.
- **There is also the possibility** that several sources will request service simultaneously. In this case the system must also decide which device to service first.
- **A priority interrupt** is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.
- **The system may also** determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.
- **Higher-priority interrupt** levels are assigned to requests which, if delayed or interrupted, could have serious consequences.
- **Devices with high speed** transfers such as magnetic disks are given high priority, and slow devices such as keyboards receive low priority.
- **When two devices** interrupt the computer at the same time, the computer services the device, with the higher priority first.
- **Establishing the priority** of simultaneous interrupts can be done by software or hardware.
- **A polling procedure** is used to identify the highest-priority source by software means.
- **In this method** there is one common branch address for all interrupts.
- **The program** that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence. The order in wJ;tich they are tested determines the priority of each interrupt.
- **The highest-priority** source is tested first, and if its interrupt signal is on, control branches to a service routine for this source.
- **Otherwise**, the next-lower-priority source is tested, and so on. Thus the initial service routine for all interrupts consists of a program that tests the interrupt sources in sequence and branches to one of many possible service routines.
- **The particular** service routine reached belongs to the highest-priority device among all devices that interrupted the computer.
- **The disadvantage** of the software method is that if there are many interrupts, the time required to poll them can exceed the time available to service the I/O device. In this situation a hardware priority-interrupt unit can be used to speed up the operation.
- **A hardware priority-interrupt** unit functions as an overall manager in an interrupt system environment.
- **It accepts interrupt** requests from many sources, determines which of the incoming requests has the highest priority, and issues an interrupt request to the computer based on this determination.
- **To speed up the operation**, each interrupt source has its own interrupt vector to access its own service routine directly. Thus no polling is required because all the decisions are established by the hardware priority-interrupt unit.
- **The hardware priority function** can be established by either a serial or a parallel connection of interrupt lines. The serial connection is also known as the daisychaining method.

8)Describe about Asynchronous Data Transfer ?

The internal operations in an individual unit of a digital system are synchronized using clock pulse. It means clock pulse is given to all registers within a unit. And all data transfer among internal registers occurs simultaneously during the occurrence of the clock pulse. Now, suppose any two units of a digital system are designed independently, such as CPU and I/O interface.

If the registers in the I/O interface share a common clock with CPU registers, then transfer between the two units is said to be synchronous. But in most cases, the internal timing in each unit is independent of each other, so each uses its private clock for its internal registers. In this case, the two units are said to be asynchronous to each other, and if data transfer occurs between them, this data transfer is called **Asynchronous Data Transfer**.

But, the Asynchronous Data Transfer between two independent units requires that control signals be transmitted between the communicating units so that the time can be indicated at which they send data. These two methods can achieve this asynchronous way of data transfer:

- **Strobe control:** A strobe pulse is supplied by one unit to indicate to the other unit when the transfer has to occur.
- **Handshaking:** This method is commonly used to accompany each data item being transferred with a control signal that indicates data in the bus. The unit receiving the data item responds with another signal to acknowledge receipt of the data.

The strobe pulse and handshaking method of asynchronous data transfer is not restricted to I/O transfer. They are used extensively on numerous occasions requiring the transfer of data between two independent units. So, here we consider the transmitting unit as a source and receiving unit as a destination.

For example, the CPU is the source during output or write transfer and the destination unit during input or read transfer.

Therefore, the control sequence during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination.

So, while discussing each data transfer method asynchronously, you can see the control sequence in both terms when it is initiated by source or by destination. In this way, each data transfer method can be further divided into parts, source initiated and destination initiated.

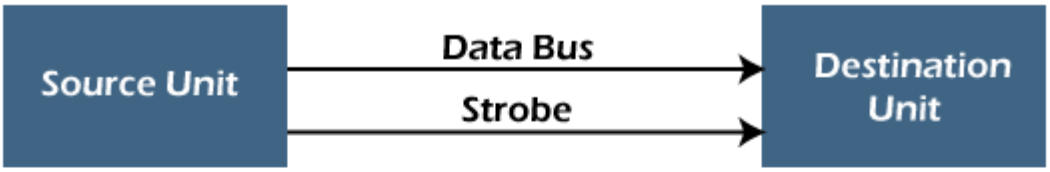
9)Explain about the methods of Asynchronous Data Transfer?

The asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate when they send the data. Thus, the two methods can achieve the asynchronous way of data transfer.

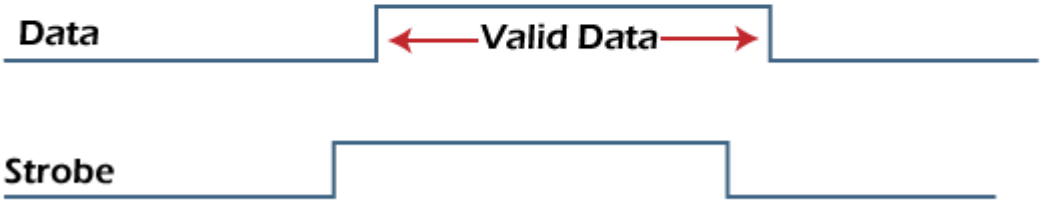
1. Strobe Control Method

The Strobe Control method of asynchronous data transfer employs a single control line to time each transfer. This control line is also known as a strobe, and it may be achieved either by source or destination, depending on which initiate the transfer.

a. **Source initiated strobe:** In the below block diagram, you can see that strobe is initiated by source, and as shown in the timing diagram, the source unit first places the data on the data bus.



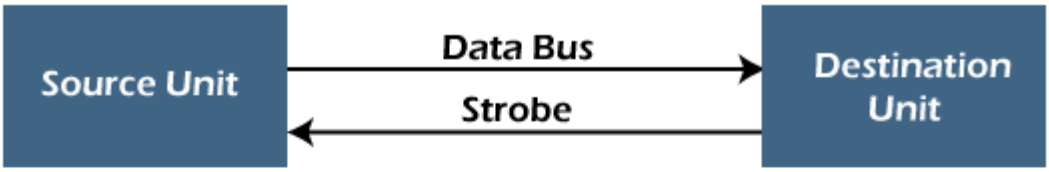
(a) Block Diagram



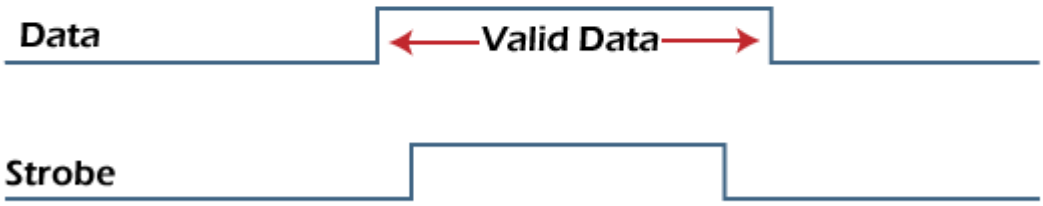
(b) Timing Diagram

After a brief delay to ensure that the data resolve to a stable value, the source activates a strobe pulse. The information on the data bus and strobe control signal remains in the active state for a sufficient time to allow the destination unit to receive the data.
The destination unit uses a falling edge of strobe control to transfer the contents of a data bus to one of its internal registers. The source removes the data from the data bus after it disables its strobe pulse. Thus, new valid data will be available only after the strobe is enabled again.
In this case, the strobe may be a memory-write control signal from the CPU to a memory unit. The CPU places the word on the data bus and informs the memory unit, which is the destination.

b. **Destination initiated strobe:** In the below block diagram, you see that the strobe initiated by destination, and in the timing diagram, the destination unit first activates the strobe pulse, informing the source to provide the data.



(a) Block Diagram



(b) Timing Diagram

The source unit responds by placing the requested binary information on the data bus. The data must be valid and remain on the bus long enough for the destination unit to accept it. The falling edge of the strobe pulse can use again to trigger a destination register. The destination unit then disables the strobe. Finally, and source removes the data from the data bus after a determined time interval. In this case, the strobe may be a memory read control from the CPU to a memory unit. The CPU initiates the read operation to inform the memory, which is a source unit, to place the selected word into the data bus.

2. Handshaking Method

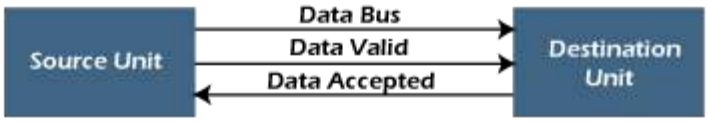
The strobe method has the disadvantage that the source unit that initiates the transfer has no way of knowing whether the destination has received the data that was placed in the bus. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has placed data on the bus.

So this problem is solved by the handshaking method. The handshaking method introduces a second control signal line that replays the unit that initiates the transfer.

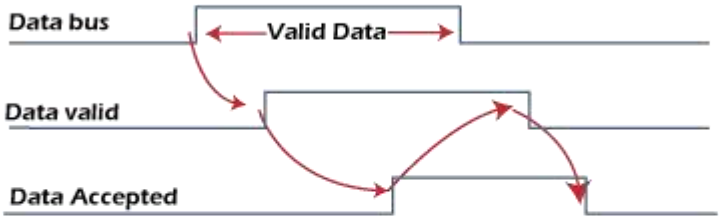
In this method, one control line is in the same direction as the data flow in the bus from the source to the destination. The source unit uses it to inform the destination unit whether there are valid data in the bus.

The other control line is in the other direction from the destination to the source. This is because the destination unit uses it to inform the source whether it can accept data. And in it also, the sequence of control depends on the unit that initiates the transfer. So it means the sequence of control depends on whether the transfer is initiated by source and destination.

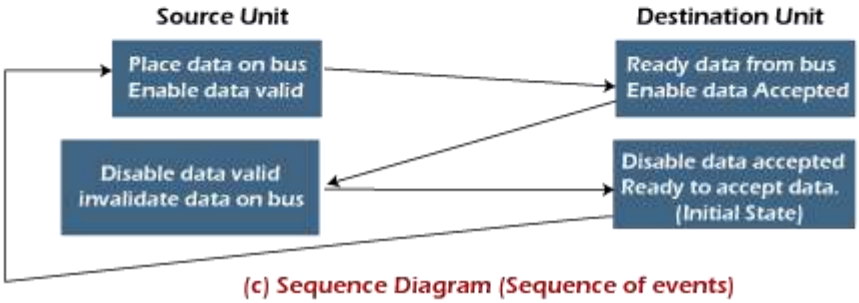
- **Source initiated handshaking:** In the below block diagram, you can see that two handshaking lines are "**data valid**", which is generated by the source unit, and "**data accepted**", generated by the destination unit.



(a) Block Diagram



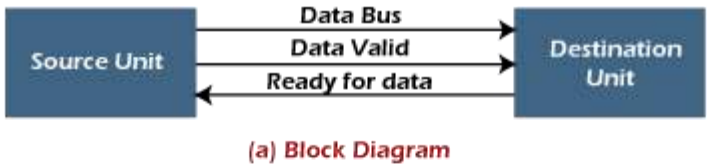
(b) Timing Diagram



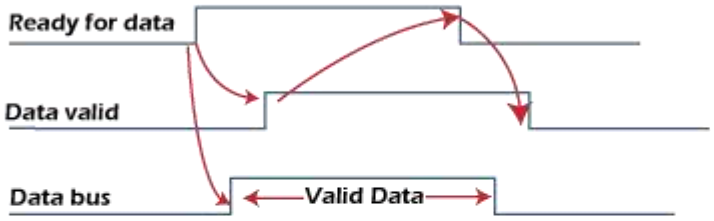
(c) Sequence Diagram (Sequence of events)

The timing diagram shows the timing relationship of the exchange of signals between the two units. The source initiates a transfer by placing data on the bus and enabling its data valid signal. The destination unit then activates the data accepted signal after it accepts the data from the bus. The source unit then disables its valid data signal, which invalidates the data on the bus. After this, the destination unit disables its data accepted signal, and the system goes into its initial state. The source unit does not send the next data item until after the destination unit shows readiness to accept new data by disabling the data accepted signal.
This sequence of events described in its sequence diagram, which shows the above sequence in which the system is present at any given time.

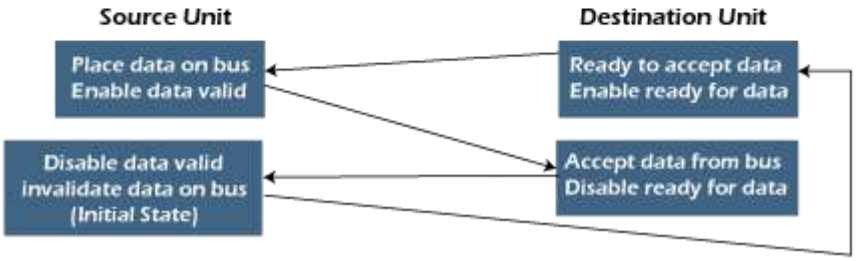
- **Destination initiated handshaking:** In the below block diagram, you see that the two handshaking lines are "**data valid**", generated by the source unit, and "**ready for data**" generated by the destination unit. Note that the name of signal data accepted generated by the destination unit has been changed to ready for data to reflect its new meaning.



(a) Block Diagram



(b) Timing Diagram



(c) Sequence Diagram (Sequence of events)

The destination transfer is initiated, so the source unit does not place data on the data bus until it receives a ready data signal from the destination unit. After that, the handshaking process is the same as that of the source initiated.

The sequence of events is shown in its sequence diagram, and the timing relationship between signals is shown in its timing diagram. Therefore, the sequence of events in both cases would be identical.

Advantages of Asynchronous Data Transfer

Asynchronous Data Transfer in computer organization has the following advantages, such as:

- It is more flexible, and devices can exchange information at their own pace. In addition, individual data characters can complete themselves so that even if one packet is corrupted, its predecessors and successors will not be affected.
- It does not require complex processes by the receiving device. Furthermore, it means that inconsistency in data transfer does not result in a big crisis since the device can keep up with the data stream. It also makes asynchronous transfers suitable for applications where character data is generated irregularly.

Disadvantages of Asynchronous Data Transfer

There are also some disadvantages of using asynchronous data for transfer in computer organization, such as:

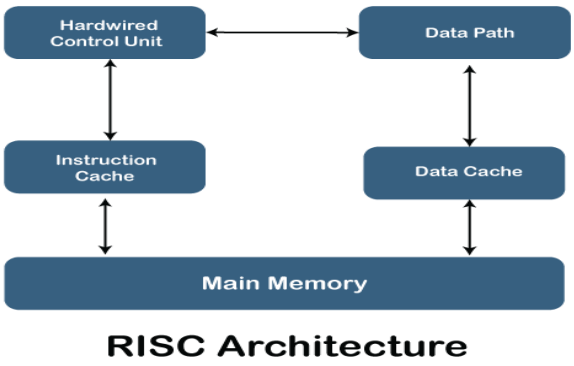
- The success of these transmissions depends on the start bits and their recognition. Unfortunately, this can be easily susceptible to line interference, causing these bits to be corrupted or distorted.

- A large portion of the transmitted data is used to control and identify header bits and thus carries no helpful information related to the transmitted data. This invariably means that more data packets need to be sent.

UNIT V

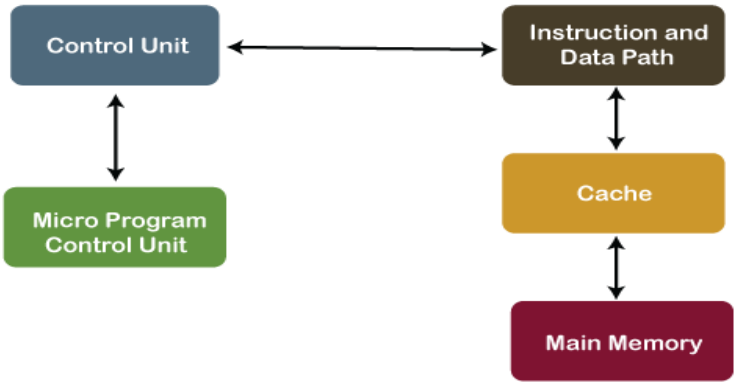
SHORT ANSWERS:

1)Draw architecture of RISC?



RISC Architecture

2)Draw architecture of CISC?



CISC Architecture

3)What is parallel processing?

- *Parallel processing* is a term used to denote a large class of techniques that are used to provide simultaneous *data-processing tasks* for the purpose of increasing the computational speed of a computer system.

4)What are the variety of parallel processing?

There are a variety of ways that parallel processing can be classified. it can beconsidered from the

- Internal organization of the processors
- Interconnection structure between processors
- The flow of information through the system

5)What id hardware interlocks?

*A*n interlock is a circuit that detects instructions whosesource operands are destinations of instructions farther up in the pipeline.
This approach maintains the program sequence by using hardware toinsert the required delays.

6)Define Delayed load?

the compiler for such computers is designed to detect a dataconflict and reorder the instructions as necessary to delay the loading of the conflicting data by inserting no-operation instructions.

7)Define Branch prediction?

A pipeline with branch prediction uses some additional logic toguess the outcome of a conditional branch instruction before it is executed.

8)What are the three types of instruction?

There are three types of instructions:

- The data manipulation instructions: operate on data in processor registers
- The data transfer instructions:
- The program control instructions

9)What are the four instruction in pipeline?

Consider the operation of the following four instructions:

1. LOAD: R1 □ M[address 1]
2. LOAD: R2 □ M[address 2]
3. ADD: R3 □ R1 +R2
4. STORE: M[address 3] □ R3

10)Define Crossbar Switch?

The crossbar switch organization consists of a number of crosspoints that are placed at intersections between processor buses and memory module paths. Figure 13-4 shows a crossbar switch interconnection between four CPUs and four memory modules

LONG ANSWERS:

1)Difference between the RISC and CISC Processors?

RISC	CISC
It is a Reduced Instruction Set Computer.	It is a Complex Instruction Set Computer.
It emphasizes on software to optimize the instruction set.	It emphasizes on hardware to optimize the instruction set.
It is a hard wired unit of programming in the RISC Processor.	Microprogramming unit in CISC Processor.
It requires multiple register sets to store the instruction.	It requires a single register set to store the instruction.
RISC has simple decoding of instruction.	CISC has complex decoding of instruction.
Uses of the pipeline are simple in RISC.	Uses of the pipeline are difficult in CISC.
It uses a limited number of instruction that requires less time to execute the instructions.	It uses a large number of instruction that requires more time to ex instructions.
It uses LOAD and STORE that are independent instructions in the register-to-register a program's interaction.	It uses LOAD and STORE instruction in the memory-to-memory intera program.
RISC has more transistors on memory registers.	CISC has transistors to store complex instructions.
The execution time of RISC is very short.	The execution time of CISC is longer.
RISC architecture can be used with high-end applications like telecommunication, image processing, video processing, etc.	CISC architecture can be used with low-end applications like home au security system, etc.
It has fixed format instruction.	It has variable format instruction.
The program written for RISC architecture needs to take more space in memory.	Program written for CISC architecture tends to take less space in memor
Example of RISC: ARM, PA-RISC, Power Architecture, Alpha, AVR, ARC and the SPARC.	Examples of CISC: VAX, Motorola 68000 family, System/360, AMD Intel x86 CPUs

2) Explain about Characteristics of CISC Processor?

Following are the main characteristics of the RISC processor:

1. The length of the code is shorts, so it requires very little RAM.
2. CISC or complex instructions may take longer than a single clock cycle to execute the code.
3. Less instruction is needed to write an application.
4. It provides easier programming in assembly language.
5. Support for complex data structure and easy compilation of high-level languages.
6. It is composed of fewer registers and more addressing nodes, typically 5 to 20.
7. Instructions can be larger than a single word.
8. It emphasizes the building of instruction on hardware because it is faster to create than the software.

3) explain in detail about vector processing?

- In many science and engineering applications, the problems can be formulated interms of vectors and matrices that lend themselves to vector processing.
  Computers with vector processing capabilities are in demand in specializedapplications. e.g.

    - Long-range weather forecasting
    - Petroleum explorations
    - Seismic data analysis
    - Medical diagnosis
    - Artificial intelligence and expert systems
    - Image processing
    - Mapping the human genome

- To achieve the required level of high performance it is necessary to utilize the *fastest and most reliable hardware* and apply innovative procedures from *vector and parallel processing techniques*.

Vector Operations

- Many scientific problems require arithmetic operations on large arrays ofnumbers.
- A vector is an ordered set of a one-dimensional array of data items.
- A vector V of length n is represented as a row vector by V=[v1,v2,…,Vn].
- To examine the difference between a conventional scalar processor and a vectorprocessor, consider the following Fortran DO loop:

```
   DO 20 I = 1, 100
20       C(I) = B(I) + A(I)
```

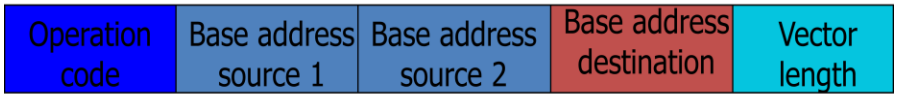
- This is implemented in machine language by the following sequence ofoperations.

```
Initialize I=020
Read A(I)Read
        B(I)
Store C(I) = A(I)+B(I)
Increment I = I + 1If I
100 go to 20 Continue
```

- A computer capable of vector processing eliminates the overhead associated withthe time it takes to fetch and execute the instructions in the program loop.

```
C(1:100) = A(1:100) + B(1:100)
```

- A possible instruction format for a vector instruction is shown in Fig. 9-11.
  - This assumes that the vector operands reside in *memory*.



- It is also possible to design the processor with a large number of *registers* andstore all operands in registers prior to the addition operation.
  - The base address and length in the vector instruction specify a group ofCPU registers.

Matrix Multiplication

- The multiplication of two n x n matrices consists of n^2 inner products or n^3 multiply-add operations.
- Consider, for example, the multiplication of two 3 x 3 matrices A and B.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

The product matrix C is a 3×3 matrix whose elements are related to the elements of A and B by the inner product:

$$c_{ij} = \sum_{k=1}^{3} a_{ik} \times b_{kj}$$

For example, the number in the first row and first column of matrix C is calculated by letting $i = 1, j = 1$, to obtain

  - c11= a11b11+ a12b21+ a13b31

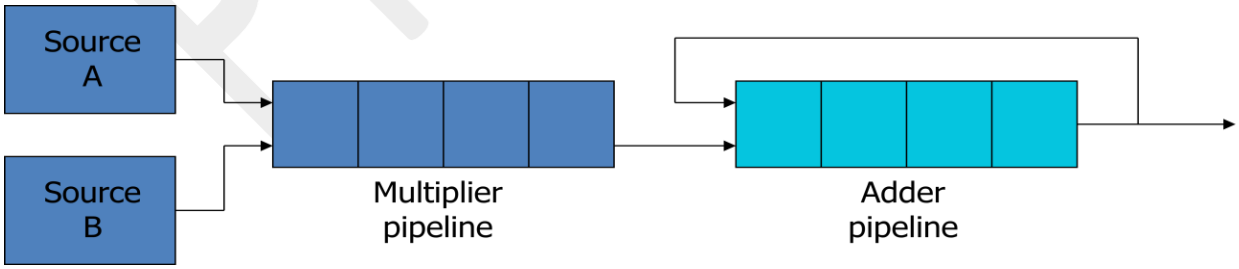
  - This requires three multiplication and (after initializing c11 to 0) threeadditions.
- In general, the inner product consists of the sum of *k* product terms of the form C = A1B1+A2B2+A3B3+…+AkBk.
  - In a typical application k may be equal to 100 or even 1000.
- The inner product calculation on a pipeline vector processor is shown in Fig. 9-12.

$$C = A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + + + +$$

$$+ A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + + + +$$

$$+ A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} + + + +$$

$$+ A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + + + +$$



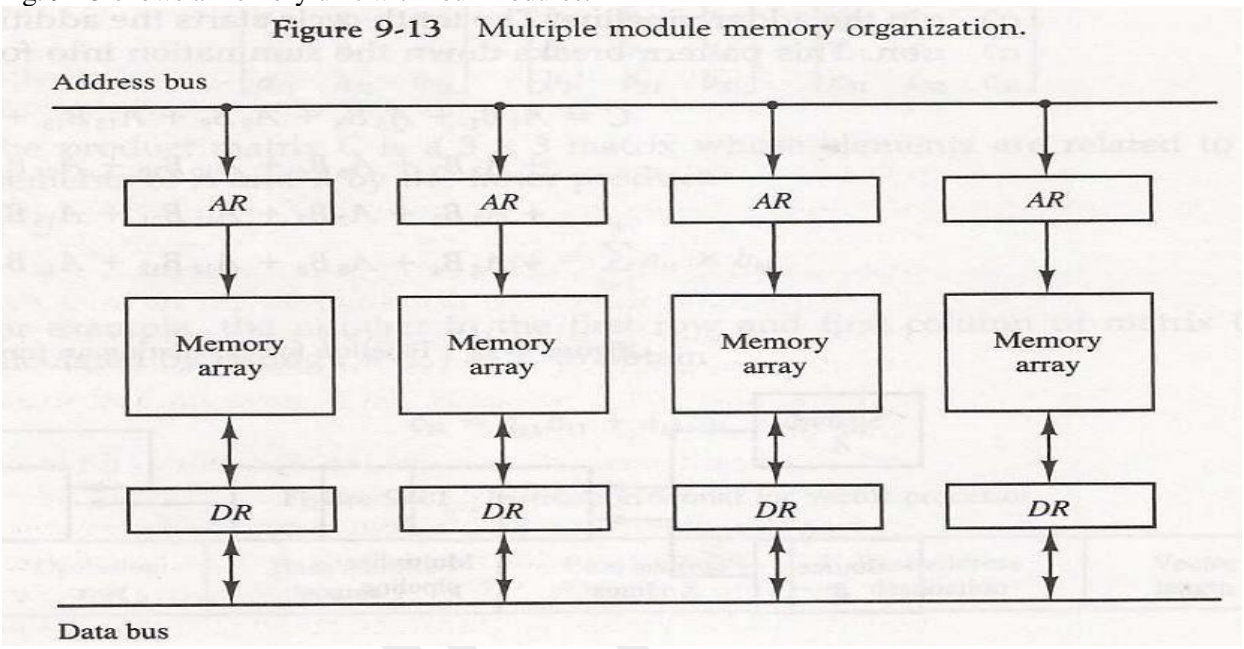
Memory Interleaving

- *Pipeline* and *vector processors* often require simultaneous access to memory fromtwo or more sources.
  - An instruction pipeline may require the fetching of an instruction and anoperand at

the same time from two different segments.
    - An arithmetic pipeline usually requires two or more operands to enter thepipeline at the same time.

- Instead of using two memory buses for simultaneous access, the memory can be partitioned into a number of modules connected to a common memory address and data buses.
    - A memory module is a memory array together with its own address and data registers.
- Fig. 9-13 shows a memory unit with four modules.

Figure 9-13 Multiple module memory organization.



- The advantage of a modular memory is that it allows the use of a techniquecalled *interleaving*.
- In an interleaved memory, different sets of addresses are assigned to differentmemory modules.
- By staggering the memory access, the effective memory cycle time can be *reduced by a factor close to the number of modules*.

Supercomputers

- A commercial computer with vector instructions and pipelined floating-pointarithmetic operations is referred to as a *supercomputer*.
    - To speed up the operation, the components are *packed tightly* together tominimize the distance that the electronic signals have to travel.
- This is augmented by instructions that process vectors and combinations of scalarsand vectors.
- A supercomputer is a computer system best known for its high computational speed,fast and large memory systems, and the extensive use of parallel processing.
    - It is equipped with *multiple functional units* and each unit has its own *pipeline* configuration.
- It is specifically optimized for the type of numerical calculations involving vectors andmatrices of floating-point numbers.
- They are limited in their use to a number of scientific applications, such *as numericalweather*

forecasting, seismic wave analysis, and space research.

- A measure used to evaluate computers in their ability to perform a given number offloating-point operations per second is referred to as *flops*.
- A typical supercomputer has a basic cycle time of 4 to 20 ns.
- The examples of supercomputer:

- Cray-1: it uses vector processing with 12 distinct functional units in parallel; a large number of registers (over 150); multiprocessor configuration (Cray X- MP and Cray Y-MP)
- Fujitsu VP-200: 83 vector instructions and 195 scalar instructions; 300 megaflops

4)Expalin about array processors?

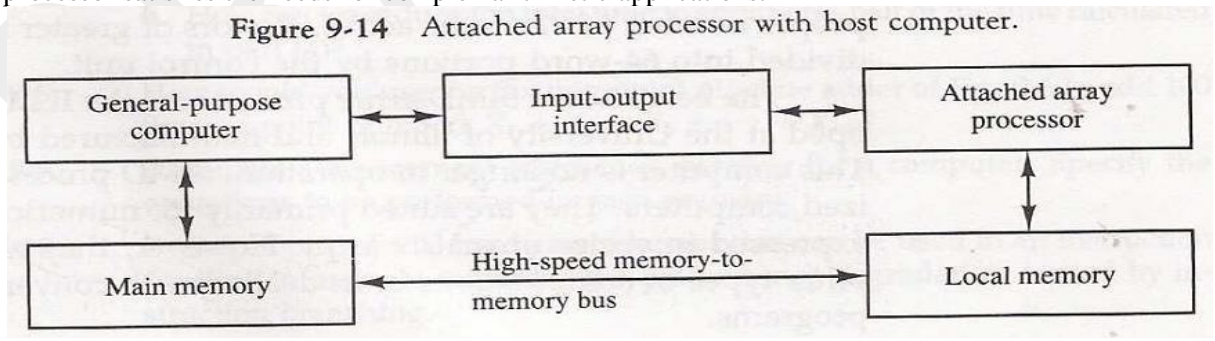
Array Processors :

- An array processor is a processor that performs computations on large arrays ofdata.
- The term is used to refer to two different types of processors.
  - Attached array processor:
    - Is an auxiliary processor.
    - It is intended to improve the performance of the host computer inspecific numerical computation tasks.
  - SIMD array processor:
    - Has a single-instruction multiple-data organization.
    - It manipulates vector instructions by means of multiple functional units responding to a common instruction.

Attached Array Processor

- Its purpose is to enhance the performance of the computer by providing vectorprocessing for complex scientific applications.
  - Parallel processing with multiple functional units
- Fig. 9-14 shows the interconnection of an attached array processor to a hostcomputer.
- The host computer is a general-purpose commercial computer and the attached processor is a back-end machine driven by the host computer. The array processor isconnected through an input-output controller to the computer and the computer treats it like an external interface.
- The data for the attached processor are transferred from main memory to a localmemory through a high-speed bus. The general-purpose computer without the attached processor serves the users that need conventional data processing. Thesystem with the attached processor satisfies the needs for complex arithmetic applications.

Figure 9-14 Attached array processor with host computer.



- For example, when attached to a VAX 11 computer, the FSP-164/MAX from Floating-Point Systems increases the computing power of the VAX to 100megaflops.
- The objective of the attached array processor is to provide *vector manipulationcapabilities* to a conventional computer at a fraction of the cost of supercomputer.

SIMD Array Processor

- An SIMD array processor is a computer with multiple processing units operating inparallel.
- A general block diagram of an array processor is shown in Fig. 9-15.
    - It contains a set of identical processing elements (PEs), each having a localmemory *M*.
    - Each PE includes an ALU, a floating-point arithmetic unit, and workingregisters.
    - Vector instructions are broadcast to all PEs simultaneously.
- Masking schemes are used to control the status of each PE during the execution ofvector instructions.
    - Each PE has a flag that is set when the PE is active and reset when the PE isinactive.

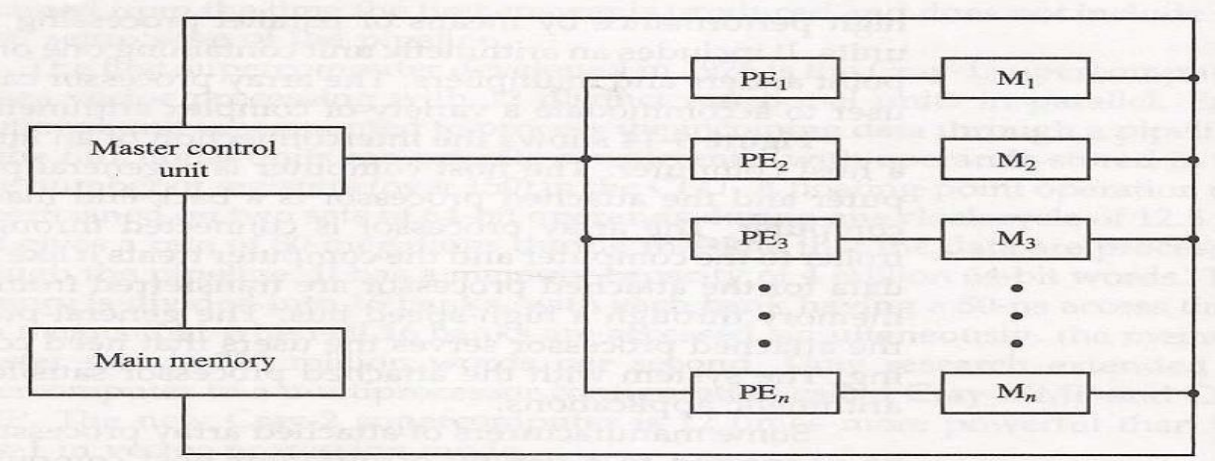


Figure 9-15 SIMD array processor organization.

- For example, the ILLIAC IV computer developed at the University of Illinois and manufactured by the Burroughs Corp.
    - Are highly specialized computers.
    - They are suited primarily for numerical problems that can be expressed invector or matrix form.

5) Explain about System Bus?

A typical system bus consists of approximately 100 signal lines. These lines are divided into three functional groups: data, address, and control. In addition, there are power distribution lines that supply power to the components.

The data lines provide a path for the transfer of data between processors and common memory. The number of data lines is usually a multiple of 8, with 16 and 32 being most common. The address lines are used to identify a memory address or any other sourceor destination, such as input or output ports. The number of address lines determines the maximum possible memory capacity in the system. For example, an address of 24 lines can access up to 224 (16 mega) words of memory. The data and address lines are terminated with three-state buffers. The address buffers are unidirectional from processor to memory. The data lines are bidirectional, allowing the transfer of data in either direction.

Data transfers over the system bus may be synchronous or asynchronous. In a synchronous bus, each data item is transferred during a time slice known in advance to both source and destination units. Synchronization is achieved by driving both units from a common clock source. An alternative procedure

is to have separate clocks of approximately the same frequency in each unit. Synchronization signals are transmitted periodically in order to keep all clocks in the system in step with each other.

In an asynchronous bus, each data item being transferred is accompanied by handshaking control signals to indicate when the data are transferred from the source and received by the destination.

The control lines provide signals for controlling the information transfer between units. Timing signals indicate the validity of data and address information. Command signals specify operations to be performed. Typical control lines include transfer signals such as memory read and write, acknowledge of a transfer, interrupt requests, bus control signals such as bus request and bus grant, and signals for arbitration procedures.

Table 13-1 lists the 86 lines that are available in the IEEE standard 796 multibus. It includes 16 data lines and 24 address lines.

TABLE 13-1 IEEE Standard 796 Multibus Signals

	Signal name
Data and address	
Data lines (16 lines)	DATA0–DATA15
Address lines (24 lines)	ADRS0–ADRS23
Data transfer	
Memory read	MRDC
Memory write	MWTC
IO read	IORC
IO write	IOWC
Transfer acknowledge	TACK
Interrupt control	
Interrupt request (8 lines)	INT0–INT7
Interrupt acknowledge	INTA
Miscellaneous control	
Master clock	CCLK
System initialization	INIT
Byte high enable	BHEN
Memory inhibit (2 lines)	INH1–INH2
Bus lock	LOCK
Bus arbitration	
Bus request	BREQ
Common bus request	CBRQ
Bus busy	BUSY
Bus clock	BCLK
Bus priority in	BPRN
Bus priority out	BPRO
Power and ground (20 lines)	

Reprinted with permission of the IEEE.

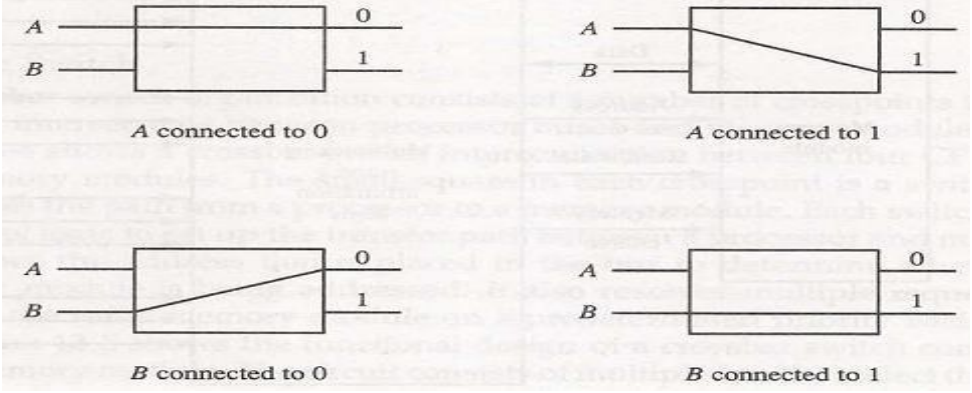
The six bus arbitration signals are used for interprocessor arbitration. These signals will be explained later after a discussion of the serial and parallel arbitration procedures.

6) Explain about Multistage Switching Network?

The basic component of a multistage network is a two-input, two-output interchange switch interchange switch. 13-6, the 2 X 2 switch has two input labeled A and B, and two outputs, labeled 0 and 1. There are control signals (not shown) associated with the switch that establish the interconnection

between the input and output terminals. The switch has the capability connecting input A to either of the outputs. Terminal B of the switch behaves in a similar fashion. The switch also has the capability to arbitrate between conflicting requests. If inputs A and B both request the same output terminal only one of them will be connected; the other will be blocked.

Figure 13-6 Operation of a 2 × 2 interchange switch.



Using the 2 X 2 switch as a building block, it is possible to build a multistage network to control the communication between a number of sources and destinations. To see how this is done, consider the binary tree shown Fig. 13-7. The two processors P1 and P2 are connected through switches to eight memory modules marked in binary from 000 through
111. The path from source to a destination is determined from the binary bits of the destination number. The first bit of the destination number determines the switch output in the first level. The second bit specifies the output of the switch in the second level, and the third bit specifies the output of the switch in the third level. For example, to connect P1 to memory 101, it is necessary to form a path from P1 to output 1 in the first-level switch.

7) Explain about Characteristics of Multiprocessors?

A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment. The term "processor" in multiprocessor can mean either a central processing unit (CPU) or an input-output processor (IOP). However, a system with a single CPU and one or more IOPs is usually not included in the definition of a multiprocessor system unless the IOP has computational facilities comparable to a CPU. As it is most commonly defined, a multiprocessor system implies the existence of multiple CPUs, although usually there will be one or more IOPs as well. Multiprocessors are classified as multiple instruction stream, multiple data MIMD stream (MIMD) systems.

There are some similarities between multiprocessor and multicomputer systems since both support concurrent operations. However, there exists an important distinction between a system with multiple computers and a system with multiple processors. Computers are interconnected with each other by means of communication lines to form a computer network. The network consists of several autonomous computers that may or may not communicate with each other. A multiprocessor system is controlled by one operating system that provides interaction between processors and all the components of the system cooperate in the solution of a problem.

Although some large-scale computers include two or more CPUs in their microprocessor overall system, it is the emergence of the microprocessor that has been the major motivation for multiprocessor systems.

Multiprocessing improves the reliability of the system so that a failure or error in one part has a

limited effect on the rest of the system. If a fault causes one processor to fail, a second processor can be assigned to perform the functions of the disabled processor. The system as a whole can continue to function correctly with perhaps some loss in efficiency.
The benefit derived from a multiprocessor organization is an improved system performance. The system derives its high performance from the fact that computations can proceed in parallel in one of two ways.

1. Multiple independent jobs can be made to operate in parallel.
2. A single job can be partitioned into multiple parallel tasks.

8)Explain about Arithmetic Pipeline?

- There are various reasons why the pipeline cannot operate at its maximumtheoretical rate.
    - Different segments may take different times to complete their sub operation.
    - It is not always correct to assume that a nonpipe circuit has the same timedelay as that of an equivalent pipeline circuit.
- There are two areas of computer design where the pipeline organization isapplicable.
    - Arithmetic pipeline
    - Instruction pipeline

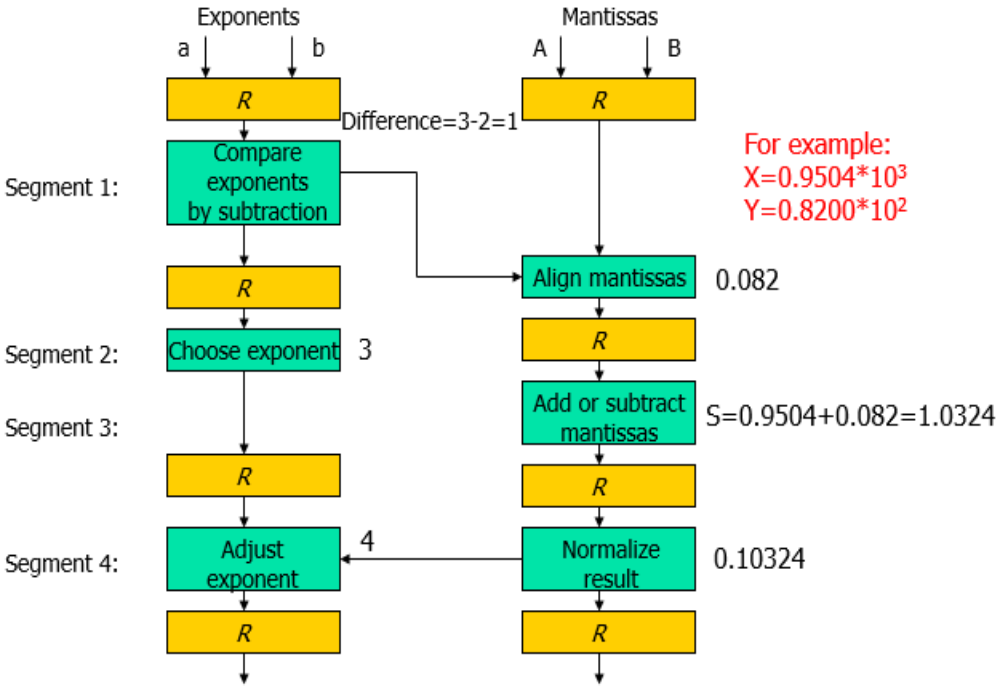
Arithmetic Pipeline

- Pipeline arithmetic units are usually found in very high speed computers
    - o Floating–point operations, multiplication of fixed-point numbers, and similarcomputations in scientific problem
- Floating–point operations are easily decomposed into suboperations as demonstratedin Sec. 10-5.
- An example of a pipeline unit for floating-point addition and subtraction is showed inthe following:
    - o The inputs to the floating-point adder pipeline are two normalized floating-point binary number

$$X = A \times 2^a$$

$$Y = B \times 2^b$$

- A and B are two fractions that represent the mantissas, a and b are the exponents.
- The floating-point addition and subtraction can be performed in four segments, asshown in Fig. 9-6.
- The suboperations that are performed in the four segments are:
    - *Compare the exponents*
        - The larger exponent is chosen as the exponent of the result.
    - *Align the mantissas*



- The exponent difference determines how many times the mantissaassociated with the smaller exponent must be shifted to the right.

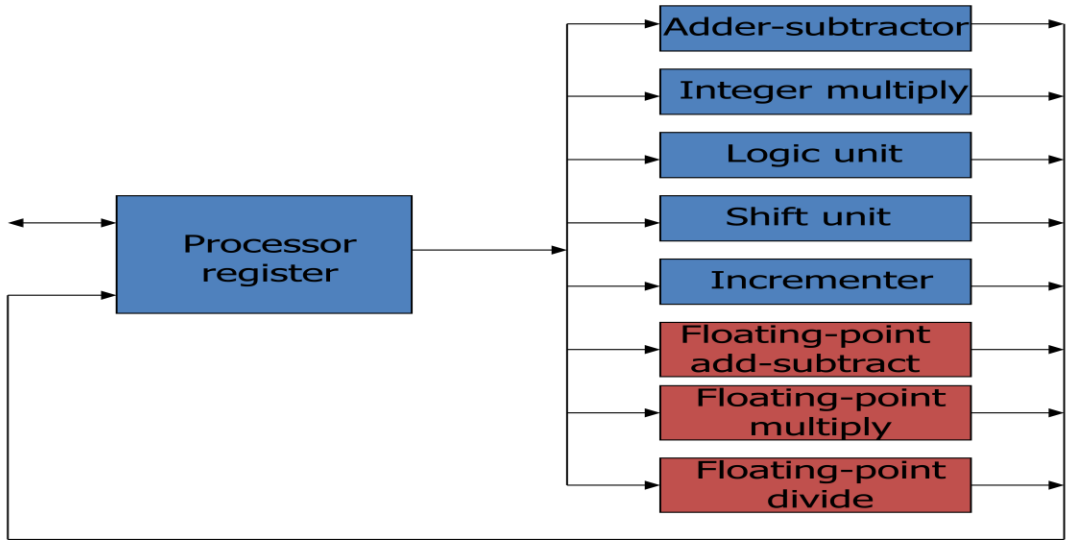
- *Add or subtract the mantissas*
- *Normalize the result*
  - When an overflow occurs, the mantissa of the sum or difference isshifted right and the exponent incremented by one.
  - If an underflow occurs, the number of leading zeros in the mantissa determines the number of left shifts in the mantissa and the numberthat must be subtracted from the exponent.

cache memory with each CPU. In addition, there is a global common memory that all CPUs can access. Information can therefore be shared among the CPUs by placing it in the common global memory.

An alternative model of microprocessor is the **distributed-memory** or **loosely coupled system**. Each processor element in a loosely coupled system has its own private local memory. The processors are tied together by a switching scheme designed to route information from one processor to another through a message-passing scheme. The processors relay program and data to other processors in packets. Loosely coupled systems are most efficient when the interaction between tasks is minimal, whereas tightly coupled systems can tolerate a higher degree of interaction between tasks.

9) Expalin about Parallel Processing?

- *Parallel processing* is a term used to denote a large class of techniques that are used to provide simultaneous *data-processing tasks* for the purpose of increasing the computational speed of a computer system.
- The purpose of parallel processing is to *speed up the computer processing capability* and *increase its throughput,* that is, the amount of processing that can be accomplished during a given interval of time.
- The amount of hardware increases with parallel processing, and with it, the cost of the system increases.
- Parallel processing can be viewed from various levels of complexity.
    - At the lowest level, we distinguish between parallel and serial operations by the type of registers used. e.g. shift registers and registers with parallel load
    - At a higher level, it can be achieved by having a multiplicity of functional unitsthat perform identical or different operations simultaneously.
- Fig. below shows one possible way of separating the execution unit into eightfunctional units operating in parallel.
    - A multifunctional organization is usually associated with a complex controlunit to coordinate all the activities among the various components.


Adder-subtractor
Integer multiply
Logic unit
Shift unit
Incrementer
Floating-point add-subtract
Floating-point multiply
Floating-point divide
Processor register

`The operands in the registers are applied to one of the units depending on the operation specified by the instruction associated with the operands. The operation performed in each functional unit is indicated in each block of the diagram. The adder and integer multiplier perform the arithmetic operations with integer numbers. The floating-point operations are separated into three circuits operating in parallel. The logic, shift, and increment operations can be performed concurrently on different data. All units are independent of each other, so one number can be shifted while another number is being incremented. A multifunctional organization is usually associated with a complex control unit to coordinate all the activities among the various components.

- There are a variety of ways that parallel processing can be classified. it can beconsidered from the
    - Internal organization of the processors
    - Interconnection structure between processors
    - The flow of information through the system

One classification introduced by M.]. Flynn considers the organization of a computer system by the number of instructions and data items that are manipulated simultaneously. The normal operation of a computer is to fetch instructions from memory and execute them in the processor.

The sequence of instructions read from memory constitutes an *instruction stream.* The operations performed on the data in the processor constitute a *data stream*. Parallelprocessing may occur in the instruction stream, in the datastream, or in both. Flynn's classification divides computers into four major
groups as follows:

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data stream (SIMD)
- Multiple instruction stream, single data stream (MISD)
- Multiple instruction stream, multiple data stream (MIMD)

Single instruction stream, single data stream (SISD)

- Represents the organization of a single computer containing a control unit, aprocessor unit, and a memory unit.
- Instructions are executed *sequentially* and the system may or may not have internalparallel processing capabilities.
- parallel processing may be achieved by means of *multiple functional units* or by *pipeline processing*.

Single instruction stream, multiple data stream (SIMD)

- Represents an organization that includes many processing units under thesupervision of a common control unit.
- All processors receive the same instruction from the control unit but operate ondifferent items of data.
- The shared memory unit must contain *multiple modules* so that it can communicatewith all the processors simultaneously.

Multiple instruction stream, single data stream (MISD)

- MISD structure is only of theoretical interest since no practical system has beenconstructed using this organization.

Multiple instruction stream, multiple data stream (MIMD)

- MIMD organization refers to a computer system capable of processing severalprograms at the same time. e.g. multiprocessor and multicomputer system.

10)Explain about Pipelining?

Pipelining is a technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments. The name "pipeline" implies aflow of information analogous to an industrial assembly line. It is characteristic of pipelines that several computations can be in progress in distinct segments at the same time.

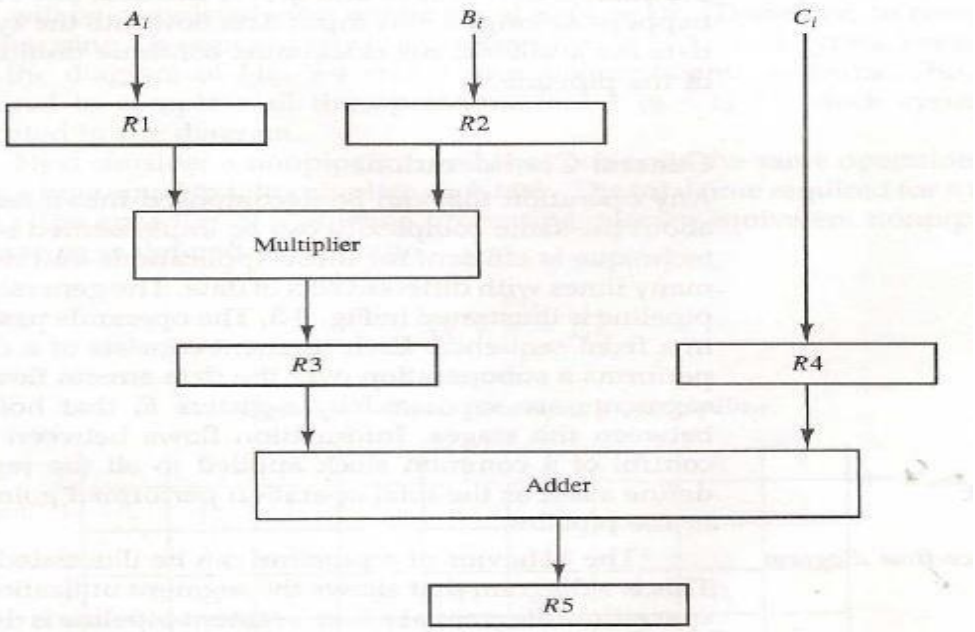
- Perhaps the simplest way of viewing the pipeline structure is to imagine that each segment consists of an *input register* followed by a *combinational circuit*.
  - The register holds the data.
  - The combinational circuit performs the suboperation in the particular segment.
- A clock is applied to all registers after *enough time* has elapsed to perform allsegment activity.
- **Example**
- The pipeline organization will be demonstrated by means of a simple example.
  - To perform the combined multiply and add operations with a stream ofnumbers
    $A_i * B_i + C_i$ for $i = 1, 2, 3, \ldots, 7$

- Each suboperation is to be implemented in a segment within a pipeline.
  - R1 ← A_i, R2 ← B_i Input A_i and B_i
  - R3 ← R1 * R2, R4 ← C_i Multiply and input C_i
  - R5 ← R3 + R4 Add C_i to product

The five registers are loaded with new data every clock pulse. The effect of each clock is shown in Table 9-1. The first clock pulse transfers A1 and 31 into R1 andR2. The second clock pulse transfers the product of R1 and R2 into R3 and C1into R4. The same clock pulse transfers A2 and B2 into R1 and R2. The third clock pulse operates on all three segments simultaneously. It places A3 and B3into R1 and R2, transfers the product of R1 and R2 into R3, transfers C2 intoR4, and places the sum of R3 and R4 into R5. It takes three clockpulses to fill up the pipe and retrieve the first output from R5. From there on, each clock produces a new output and moves the data one step down the pipeline. This happens as long as new input data flow into the system. When no more input data are available, the clock must continue until the last output emerges out of the pipeline.

- Each segment has one or two registers and a combinational circuit as shown in Fig. 9-2.

Figure 9-2 Example of pipeline processing.



- The five registers are loaded with new data every clock pulse. The effect of eachclock is shown in Table 9-1.

TABLE 9-1 Content of Registers in Pipeline Example

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$

General considerations

- Any operation that can be decomposed into a sequence of sub operations of aboutthe *same complexity* can be implemented by a pipeline processor.
- The general structure of a four-segment pipeline is illustrated in Fig. 9-3.
- We define a *task* as the total operation performed going through all the segments inthe pipeline.
- The behavior of a pipeline can be illustrated with a *space-time* diagram.
    - It shows the segment utilization as a function of time.

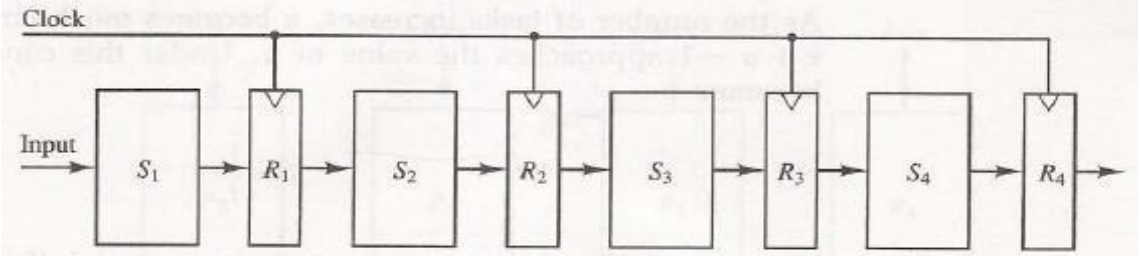


Figure 9-3 Four-segment pipeline.

- The space-time diagram of a four-segment pipeline is demonstrated in Fig. 9-4.
- Where a *k*-segment pipeline with a clock cycle time *t*p is used to execute *n* tasks.
  - The first task T1 requires a time equal to *kt*p to complete its operation.
  - The remaining *n*-1 tasks will be completed after a time equal to (*n*-1)tp
  - Therefore, to complete *n* tasks using a *k*-segment pipeline requires *k*+(*n*-1)clock cycles.
- Consider a nonpipeline unit that performs the same operation and takes a time equalto *t*n to complete each task.
  - The total time required for *n* tasks is *nt*n.

Figure 9-4 Space-time diagram for pipeline.

	1	2	3	4	5	6	7	8	9	→ Clock cycles
Segment: 1	T_1	T_2	T_3	T_4	T_5	T_6				
2		T_1	T_2	T_3	T_4	T_5	T_6			
3			T_1	T_2	T_3	T_4	T_5	T_6		
4				T_1	T_2	T_3	T_4	T_5	T_6	

- The *speedup of a pipeline processing* over an equivalent nonpipeline processing isdefined by the ratio $S = nt_n/(k+n-1)t_p$.
- If *n* becomes much larger than *k*-1, the speedup becomes $S = t_n/t_p$.
- If we assume that the time it takes to process a task is the same in the pipeline andnonpipeline circuits, i.e.,$t_n = kt_p$, the speedup reduces to $S=kt_p/t_p=k$.
- This shows that the theoretical maximum speedup that a pipeline can provide is *k*,where *k* is the number of segments in the pipeline.

- To duplicate the theoretical speed advantage of a pipeline process by means of multiple functional units, it is necessary to construct *k* identical units that will beoperating in parallel.
- This is illustrated in Fig. 9-5, where four identical circuits are connected in parallel.
- Instead of operating with the *input data in sequence* as in a pipeline, the parallel circuits accept four input data items *simultaneously* and perform four tasks at thesame time.

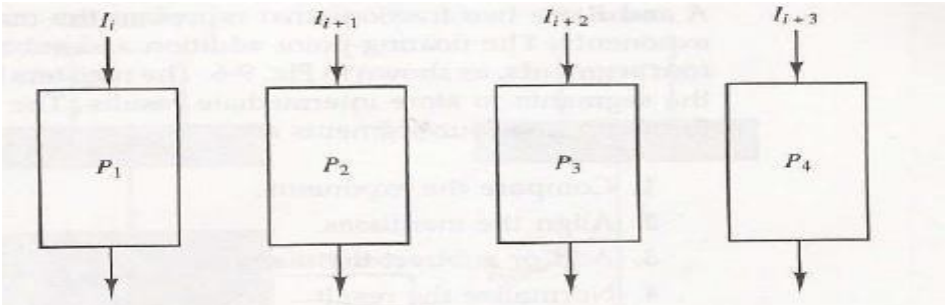


Figure 9-5 Multiple functional units in parallel.

- We consider parallel processing under the following main topics:
    - Pipeline processing
        - Is an implementation technique where arithmetic sub operations orthe phases of a computer instruction cycle overlap in execution.
    - Vector processing
        - Deals with computations involving large vectors and matrices.
    - Array processing
        - Perform computations on large arrays of data.

output 0 in the second-level switch, and output 1 in the third-level switch. It is clear that either P1 or P; can be connected to any one of the eight memories. Certain request patterns, however, cannot be satisfied simultaneously. For example, if P1 is connected to one of the destinations 000 through 011, P2 can be connected to only one of the destinations 100 through 111.

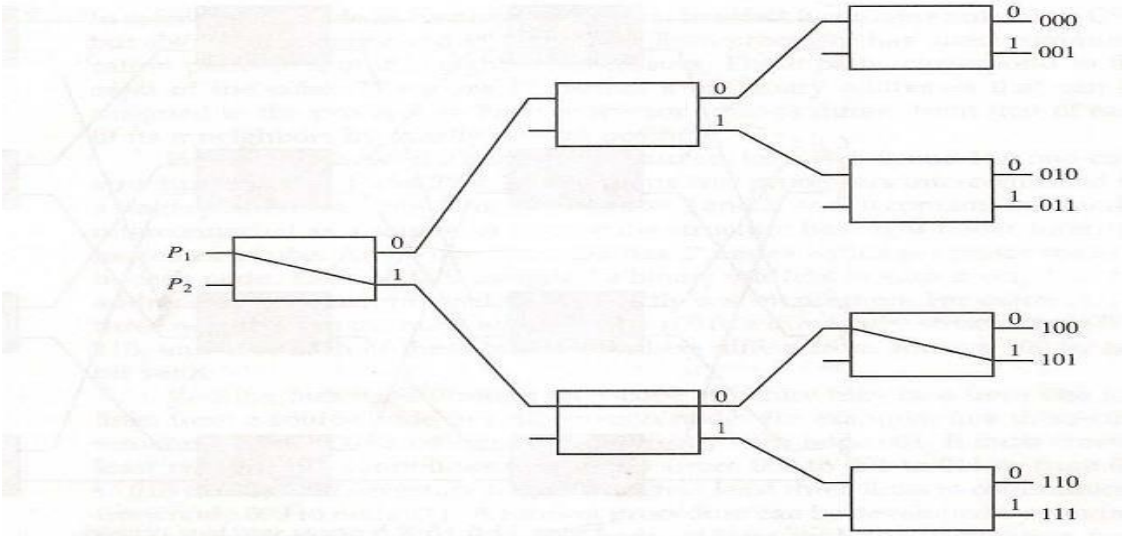


Figure 13-7 Binary tree with 2 × 2 switches.

Many different topologies have been proposed for multistage switching networks to control processor-memory communication in a tightly coupled multiprocessor system or to control the communication between the processing elements in a loosely coupled system. One such topology is the **omega network switching network** shown in Fig. 13-8. In this configuration, there is exactly one path from each source to any particular destination. Some request patterns, however, cannot be connected simultaneously. For example, any two sources cannot be connected simultaneously to destinations 000 and 001.

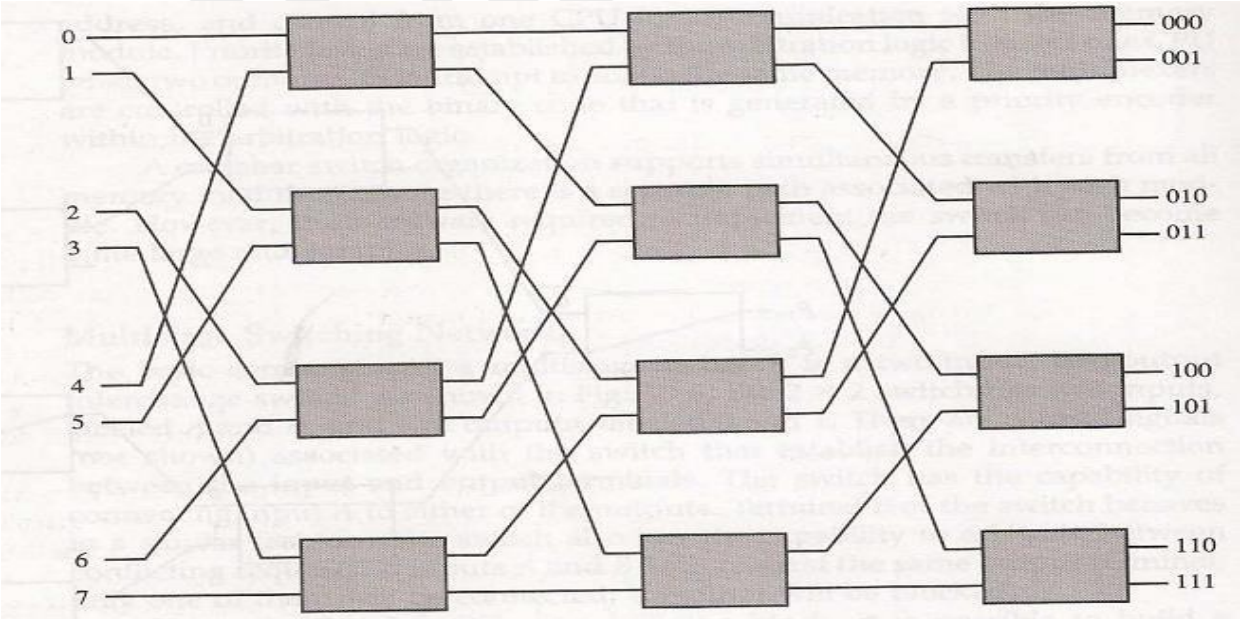


Figure 13-8 8 × 8 omega switching network.

A particular request is initiated in the switching network by the source, which sends a 3-bit pattern representing the destination number. As the binary pattern moves through the network, each level examines a different bit to determine the 2 X 2 switch setting. Level 1 inspects the most significant bit, level 2 inspects the middle bit, and level 3 inspects the least significant bit. When the request arrives on either input of the 2 x 2 switch, it is routedto the upper output if the specified bit is 0 or to the lower output if the bit is 1.

In a tightly coupled multiprocessor system, the source is a processor and the destination is a memory module. The first pass through the network sets up the path. Succeeding passes are used to transfer the address into memory and then transfer the data in either direction, depending on whether the request is a read or a write. In a loosely coupled multiprocessor system, both the source and destination are processing elements. After the path is established, the source processor transfers a message to the destination processor.