CHAPTER ONE

INTRODUCTION TO COMPUTER ARCHITECTURE

1.1 COMPUTER ARCHITECTURE: -Is a blueprint for design and implementation of a computer system. It provides the functional details and behaviour of a computer system and comes before computer organization. Computer architecture deals with 'What to do?' The architectural design of a computer system is concerned with the specifications of the various functional modules, such as processors and memories, and structuring them together into a computer system.

Computer Organization is how operational parts of a computer system are linked together. It implements the provided computer architecture. Computer organization deals with 'How to do?'

The **Instruction Set Architecture (ISA)** is the part of the processor that is visible to the programmer or compiler writer. The ISA serves as the boundary between software and hardware. The instruction set architecture (or **ISA**) is one of the most **important design** issues that a **CPU designer** must get right from the start. Features like caches, pipelining, superscalar implementation, etc.,

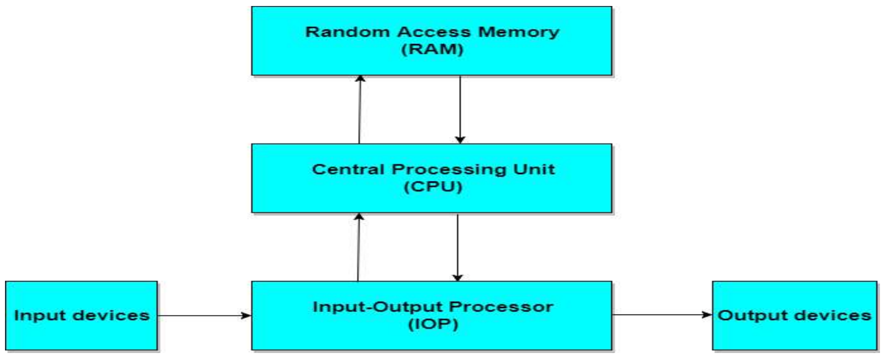
1.2 TWO BASIC TYPES OF COMPUTER ARCHITECTURE

1. **Von Neumann architecture**
2. **Harvard architecture**

1.2.1 The Von Neumann architecture describes a general framework, or structure, that a computer's hardware, programming, and data should follow. Although other structures for computing have been devised and implemented, the vast majority of computers in use today operate according to the Von Neumann architecture. Von Neumann envisioned the structure of a computer system as being composed of the following components:

1. **ALU:** The **Arithmetic Logic Unit** that performs the computer's computational and logical functions.
2. **RAM:** Memory; more specifically, the computer's main, or fast, memory, also known as **Random Access Memory (RAM)**.

3. **Control Unit**: This is a component that directs other components of the computer to perform certain actions, such as directing the fetching of data or instructions from memory to be processed by the ALU; and
4. **Man-machine interfaces;** i.e. input and output devices, such as keyboard for input and display monitor for output.



Random Access Memory (RAM)
Central Processing Unit (CPU)
Input devices
Input-Output Processor (IOP)
Output devices

Block diagram of a Digital Computer

An example of computer architecture base on the Von Neumann architecture is the desktop **personal computer**.

1.2.2 Harvard architecture uses physically separate **storage** and **signal pathways** for their instructions and data. The term originated from the **Harvard Mark I** and the data in relay latches (23- digits wide).

In a computer with Harvard architecture, the CPU can read both an instruction and data from memory at the same time, leading to double the memory bandwidth.

Microcontroller (single-chip microcomputer)-based computer systems and **DSP** (Digital Signal Processor)-based computer systems are examples of Harvard architecture.

CHAPTER TWO

BASICS OF DIGITAL COMPONENTS

2.1 INTEGRATED CIRCUIT (IC):- Complex digital circuits are constructed with integrated circuits. **IC** is a small silicon semiconductor crystal, called a chip, containing the electronic components for the digital gates. The various gates are interconnected inside the chip to form the required circuit. The chip is mounted in a ceramic or plastic container and the connections are welded to the external pins to form an IC. The number of pins of IC varies from 14 to several thousand. Each pin is identified by a unique number printed on its body.

Categories of Integrated Circuits

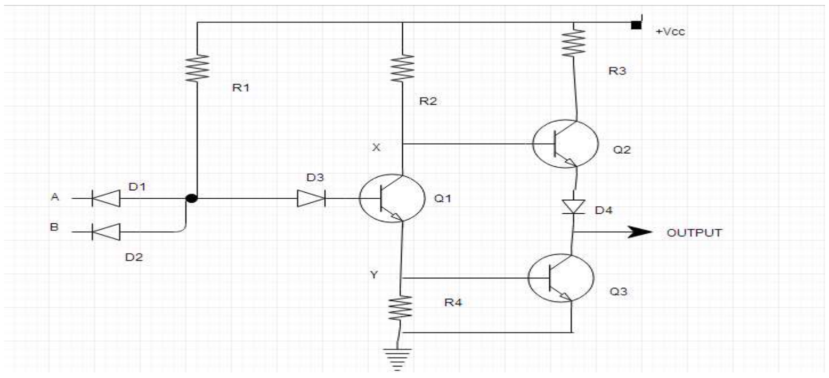
1. **SSI (Small Scale Integration Device):-** It contains several independent gates in a single package. The inputs and outputs of gates are connected directly to the pins in the package. The number of gates is usually less than 10.
2. **MSI (Medium Scale Integration Device):-** It contains 10 to 200 gates in a single package. They perform elementary digital functions such as decoders, adders, registers.
3. **LSI (Large Scale Integration Device):-** It contains gates between 200 to few thousand in a single package. They include digital systems such as processors, memory chips etc.
4. **VLSI (Very Large Scale Integration Device):-**It contains thousands of gates within a single package such as microcomputer chip.
5. **ULSI (Ultra Large Scale Integration Device):-** It contains hundreds of thousands of gates within a single package such as microcomputer chip.

2.2 DIGITAL LOGIC FAMILY:- IC's are also classified by the specific circuit technology to which they belong. The basic circuit in each technology is **NAND**, **NOR**, **NOT** gates.

The earliest logic family was **Resistor-transistor logic** which used a resistor as input and a transistor as switching device.

2.2.1 Diode-transistor logic: - is a direct ancestor of the Transistor-transistor logic, and used a diode for logic functions while a transistor for amplifying functions.

2.2.2 TTL (Transistor Transistor Logic):- It is the modified form of DTL(Diode Transistor Logic), invented in 1961 by James L Buie. The diodes were replaced by transistor to improve the circuit operation. It is called transistor-transistor logic because transistor performs both the logic function and the amplifying function.



Two inputs TTL NAND gate with one output.

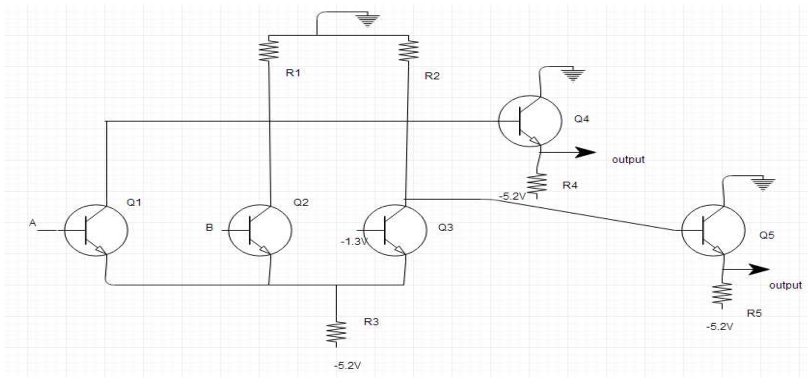
Advantages:

- TTL circuits are fast.
- Low propagation delay.
- Power dissipation does not depend upon the frequency.

Some other important points:

- TTL circuits consume more power when at rest (i.e. not being used), but as mentioned earlier, that the power dissipation does not depend on frequency, hence the power speed does not increase with clock speed as fast as for other CMOS devices.
- Compared to ECL circuits, TTL uses less power but is comparatively slower.
- TTL is less prone to damage due to electrostatic discharge than the early CMOS devices.
- Before VLSI devices, TTL circuits were used in the construction of processors for mini-computers and mainframe processors.

2.2.3 ECL (Emitter Coupled Logic):- It provides highest speed digital circuits. It is used in systems such as supercomputers and signal processors where high speed is required. ECL uses overdriven BJT (Bipolar junction Transistors) in its circuit.



Two input ECL gate.

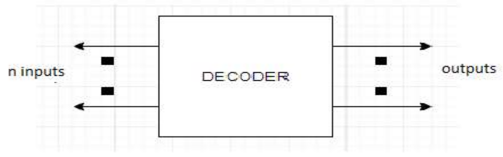
Advantages:

- ECL is the fastest logic family.
- Propagation delay is very less.
- Noise margin is low.

Some other Important points:

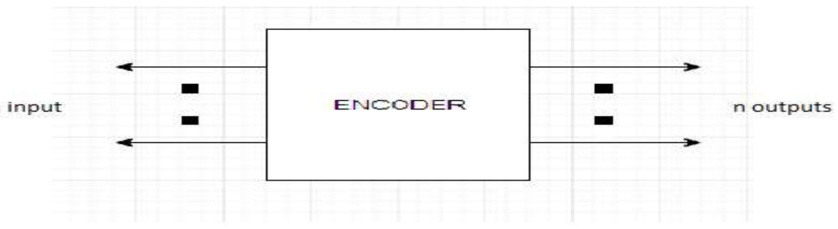
- In ECL circuit, each gate continuosly draws current even when inactive, hence power consumption is more as compared to other logic families.
- The large current requirement of ECL is contant and does not depend on the state of the circuit, which is the reason for lower power noise.

2.3 DECODERS:- A decoder is a combinational circuit that converts binary information from n coded inputs to 2^n outputs. Commercial decoders include one or more enable (E) inputs to control the operation of circuit. The decoder is enabled when E is equal to 1 and disabled when E is equal to 0.



- Used in code converters.
- Used to implement Boolean functions.

2.4 ENCODERS:- An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n input lines and n output lines. It converts octal input to binary digits.



Types of Encoders

- Priority encoders.
- Decimal to BCD encoder.
- Octal to binary encoder.
- Hexadecimal to binary encoder.

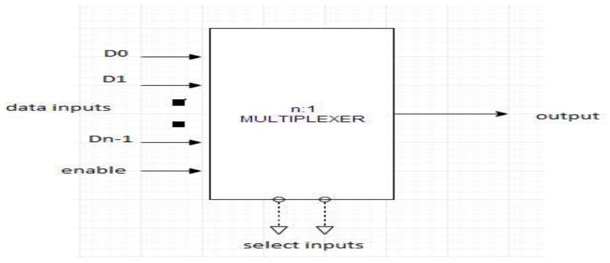
Advantages:

- Highly reliable and accurate.
- Low-cost feedback.
- High resolution.
- Integrated electronics.
- Fuses optical and digital technology.
- Can be incorporated into existing applications.
- Compact size.

Disadvantages:

- Subject to magnetic or radio interference (Magnetic Encoders).
- Direct light source interference (Optical Encoders).
- Susceptible to dirt, oil and dust contaminates.

2.5 MULTIPLEXER:- A multiplexer is a combinational circuit that receives binary information from one of the 2^n input lines and directs it to a single output line.



Advantages:

- It reduces number of wires.
- It reduces circuit complexity and cost.
- It simplifies logic design.
- We can implement many combinational circuits using MUX.
- It does not need k-maps and simplification

Disadvantages:

The following disadvantages arise while using multiplexer chips to expand I/O ports on an Arduino:

- Added delays in switching ports.
- Added delays in I/O signals propagating through the multiplexer.
- Limitations on which ports can be used simultaneously.
- Added firmware complexity to handle switching ports.
- Extra I/O ports required to control the multiplexer.

CHAPTER THREE

ARCHITECTURE OF COMPUTER SYSTEM

3.1 COMPUTER SYSTEM:- Computer is an electronic machine that makes performing any task very easy. In computer, the CPU executes each instruction provided to it, in a series of steps, this series of steps is called **Machine Cycle**, and is repeated for each instruction. One machine cycle involves *fetching of instruction*, *decoding the instruction*, *transferring the data*, *executing the instruction*.

Computer system has five basic units that help the computer to perform operations, which are given below:

1. Input Unit
2. Output Unit
3. Storage Unit
4. Arithmetic Logic Unit
5. Control Unit

Input Unit:- Input unit connects the external environment with internal computer system. It provides data and instructions to the computer system. Commonly used input devices are *keyboard*, *mouse*, *magnetic tape* etc.

Input unit performs following tasks:

- Accept the data and instructions from the outside environment.
- Convert it into machine language.
- Supply the converted data to computer system.

Output Unit:- It connects the internal system of a computer to the external environment. It provides the results of any computation, or instructions to the outside world. Some output devices are *printers*, *monitor* etc.

Storage Unit:- This unit holds the data and instructions. It also stores the intermediate results before these are sent to the output devices. It also stores the data for later use.

The storage unit of a computer system can be divided into two categories:

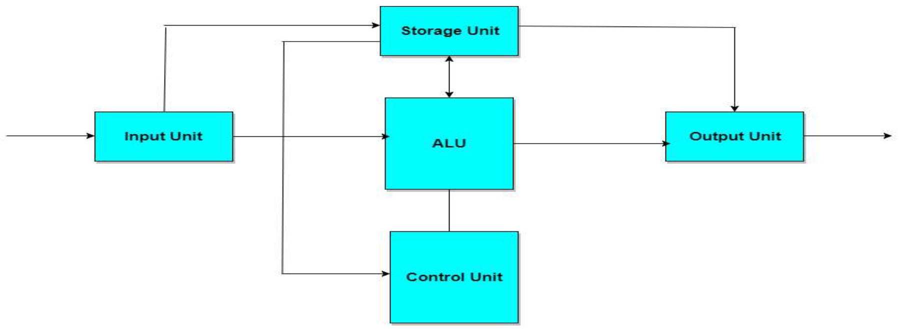
- **Primary Storage**: This memory is used to store the data which is being currently executed. It is used for temporary storage of data. The data is lost, when the computer is switched off. RAM is used as primary storage memory.
- **Secondary Storage**: The secondary memory is slower and cheaper than primary memory. It is used for permanent storage of data. Commonly used secondary memory devices are hard disk, CD etc.

Arithmetic Logical Unit: - All the calculations are performed in ALU of the computer system. The ALU can perform basic operations such as addition, subtraction, division, multiplication etc. Whenever calculations are required, the control unit transfers the data from storage unit to ALU. When the operations are done, the result is transferred back to the storage unit.

Control Unit:- It controls all other units of the computer. It controls the flow of data and instructions to and from the storage unit to ALU. Thus it is also known as central nervous system of the computer.

CPU:- It is Central Processing Unit of the computer. The control unit and ALU are together known as CPU. CPU is the brain of computer system. It performs following tasks:

- It performs all operations.
- It takes all decisions.
- It controls all the units of computer.



The block diagram of a computer.

CHAPTER FOUR

LOGIC GATES USED IN DIGITAL COMPUTERS

4.1 LOGIC GATES:- Binary logic deals with binary variables and with operations that assume a logical meaning. It is used to describe, in algebraic or tabular form, the manipulation done by logic circuits called **gates**.

Gates are blocks of hardware that produce graphic symbol and its operation can be described by means of an algebraic expression. The input-output relationship of the binary variables for each gate can be represented in tabular form by a truth-table.

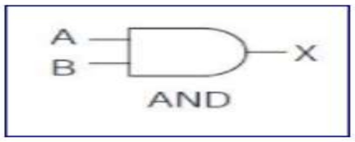
The most basic logic gates are **AND** and **inclusive OR** with multiple inputs and **NOT** with a single input.

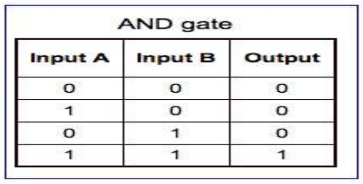
Each gate with more than one input is sensitive to either logic 0 or logic 1 input at any one of its inputs, generating the output according to its function. For example, a multi-input AND gate is sensitive to logic 0 on any one of its inputs, irrespective of any values at other inputs. The various logical gates are: AND, OR, NOT, NAND, NOR, XOR and XNOR

AND Gate:- The AND gate produces the AND logic function, that is, the output is 1 if input A and input B are both equal to 1; otherwise the output is 0.

The algebraic symbol of the AND function is the same as the **multiplication** symbol of ordinary arithmetic.

We can either use a **dot** between the variables or concatenate the variables without an operation symbol between them. AND gates may have more than two inputs, and by definition, the output is 1 if and only if all inputs are 1.



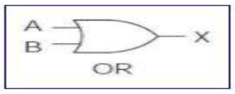

AND gate

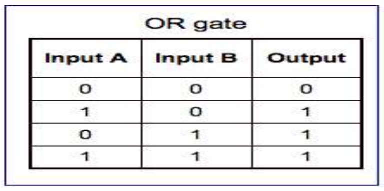
Input A	Input B	Output
0	0	0
1	0	0
0	1	0
1	1	1

OR Gate:- The OR gate produces the inclusive-OR function; that is, the output is 1 if input A or input B or both inputs are 1; otherwise, the output is 0.

The algebraic symbol of the OR function is +, similar to arithmetic **addition**.

OR gates may have more than two inputs, and by definition, the output is 1 if any input is 1.



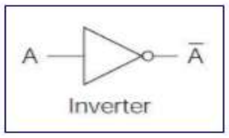


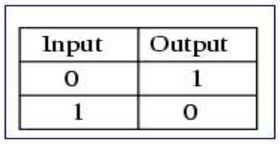
OR gate

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	1

Inverter (NOT) Gate:- The inverter circuit inverts the logic sense of a binary signal. It produces the NOT, or complement, function.

The algebraic symbol used for the logic complement is either a prime or a bar over the variable symbol.

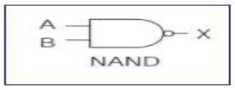


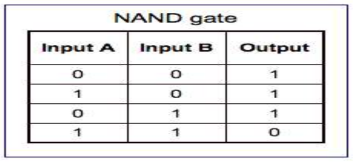


Input	Output
0	1
1	0

NAND Gate:- The NAND function is the complement of the AND function, as indicated by the graphic symbol, which consists of an AND graphic symbol followed by a small circle.

The designation NAND is derived from the abbreviation of NOT-AND.

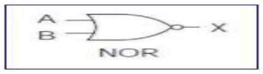


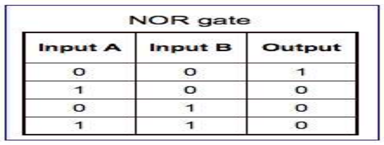


NAND gate

Input A	Input B	Output
0	0	1
1	0	1
0	1	1
1	1	0

NOR Gate:- The NOR gate is the complement of the OR gate and uses an OR graphic symbol followed by a small circle.



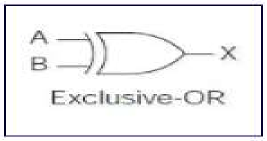


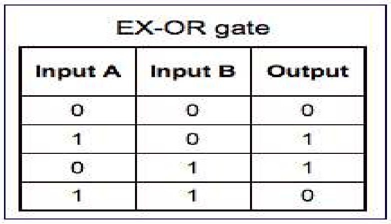
NOR gate

Input A	Input B	Output
0	0	1
1	0	0
0	1	0
1	1	0

Exclusive-OR Gate:- The exclusive-OR gate has a graphic symbol similar to the OR gate except for the additional curved line on the input side.

The output of the gate is 1 if any input is 1 but excludes the combination when both inputs are 1. It is similar to an odd function; that is, its output is 1 if an odd number of inputs are 1.

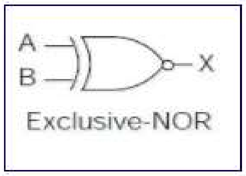


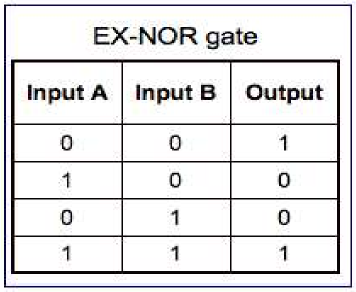

EX-OR gate

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	0

Exclusive-NOR Gate:- The exclusive-NOR is the complement of the exclusive-OR, as indicated by the small circle in the graphic symbol.

The output of this gate is 1 only if both the inputs are equal to 1 or both inputs are equal to 0.




EX-NOR gate

Input A	Input B	Output
0	0	1
1	0	0
0	1	0
1	1	1

CHAPTER FIVE

REGISTERS IN COMPUTER ARCHITECTURE

5.1 REGISTER: Register is a very fast computer memory, used to store data/instruction in-execution. A **Register** is a group of flip-flops with each flip-flop capable of storing **one bit** of information. An *n-bit* register has a group of *n flip-flops* and is capable of storing binary information of *n-bits*. The simplest register is one that consists of only flip-flops with no external gates.

These days' registers are also implemented as a register file.

Loading the Registers

The transfer of new information into a register is referred to as loading the register. If all the bits of register are loaded simultaneously with a common clock pulse than the loading is said to be done in parallel.

Register Transfer Language

The symbolic notation used to describe the micro-operation transfers amongst registers is called **Register transfer language**.

The term **register transfer** means the availability of **hardware logic circuits** that can perform a stated micro-operation and transfer the result of the operation to the same or another register.

The word **language** is borrowed from programmers who apply this term to programming languages. This programming language is a procedure for writing symbols to specify a given computational process.

Following are some commonly used registers:

1. **Accumulator**: This is the most common register, used to store data taken out from the memory.
2. **General Purpose Registers**: This is used to store data intermediate results during program execution. It can be accessed via assembly programming.
3. **Special Purpose Registers**: Users do not access these registers. These registers are for Computer system,

- **MAR:** Memory Address Register is that register that holds the address for memory unit.
- **MBR:** Memory Buffer Register stores instruction and data received from the memory and sent from the memory.
- **PC:** Program Counter points to the next instruction to be executed.
- **IR:** Instruction Register holds the instruction to be executed.

Register Transfer

Information transferred from one register to another is designated in symbolic form by means of replacement operator.

R2 ← R1 It denotes the transfer of the data from register R1 into R2.

Normally we want the transfer to occur only in predetermined control condition. This can be shown by following **if-then** statement: if (P=1) then (R2 ← R1)

Here P is a control signal generated in the control section.

Control Function

A control function is a Boolean variable that is equal to 1 or 0. The control function is shown as:

P: R2 ← R1

The control condition is terminated with a colon. It shows that transfer operation can be executed only if P=1.

Micro-Operations

The operations executed on data stored in registers are called micro-operations. A micro-operation is an elementary operation performed on the information stored in one or more registers.

Example: Shift, count, clear and load.

Types of Micro-Operations

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

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

Arithmetic Micro-Operations

Some of the basic micro-operations are addition, subtraction, increment and decrement.

Add Micro-Operation

It is defined by the following statement:

$R3 \rightarrow R1 + R2$

The above statement instructs the data or contents of register R1 to be added to data or content of register R2 and the sum should be transferred to register R3.

Subtract Micro-Operation

Let us again take an example:

$R3 \rightarrow R1 + R2' + 1$

In subtract micro-operation, instead of using minus operator we take **1's compliment** and add 1 to the register which gets subtracted, i.e **R1 - R2** is equivalent to **R3 → R1 + R2' + 1**

Increment/Decrement Micro-Operation

Increment and decrement micro-operations are generally performed by adding and subtracting 1 to and from the register respectively

$R1 \rightarrow R1 + 1$

$R1 \rightarrow R1 - 1$

Symbolic Designation	Description
R3 ← R1 + R2	Contents of R1+R2 transferred to R3.
R3 ← R1 - R2	Contents of R1-R2 transferred to R3.
R2 ← (R2)'	Compliment the contents of R2.
R2 ← (R2)' + 1	2's compliment the contents of R2.
R3 ← R1 + (R2)' + 1	R1 + the 2's compliment of R2 (subtraction).
R1 ← R1 + 1	Increment the contents of R1 by 1.
R1 ← R1 – 1	Decrement the contents of R1 by 1.

Logic Micro-Operations

These are binary micro-operations performed on the bits stored in the registers. These operations consider each bit separately and treat them as binary variables.

Let us consider the X-OR micro-operation with the contents of two registers R1 and R2.

P: R1 ← R1 X-OR R2

In the above statement we have also included a Control Function.

Assume that each register has 3 bits. Let the content of R1 be **010** and R2 be **100**. The X-OR micro-operation will be:

010 → R1

100 → R2

110 → R1 after P=1

Shift Micro-Operations

These are used for serial transfer of data. That means we can shift the contents of the register to the left or right. In the **shift left** operation the serial input transfers a bit to the right most position and in **shift right** operation the serial input transfers a bit to the left most position.

There are three types of shifts as follows:

a) Logical Shift

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

```
R1 ← she R1

R1 ← she R1
```

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

b) Circular Shift

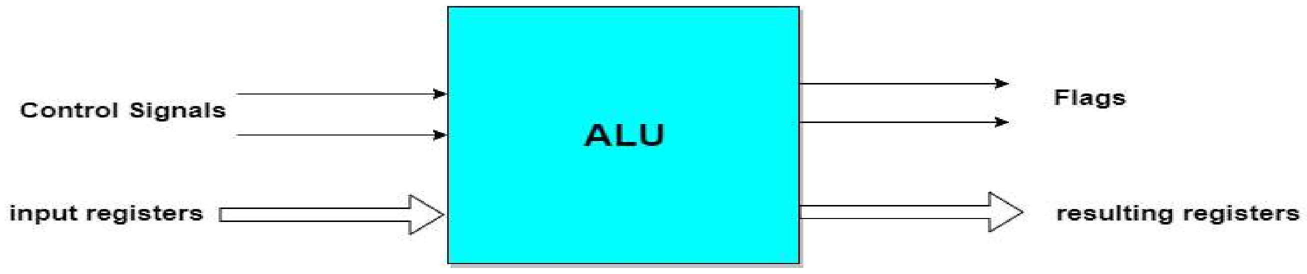
This circulates or rotates the bits of register around the two ends without any loss of data or contents. In this, the serial output of the shift register is connected to its serial input. **"cil"** and **"cir"** is used for circular shift left and right respectively.

c) Arithmetic Shift

This shifts a signed binary number to left or right. An **arithmetic shift left** multiplies a signed binary number by 2 and **shift left** divides the number by 2. Arithmetic shift micro-operation leaves the sign bit unchanged because the signed number remains same when it is multiplied or divided by 2.

Arithmetic Logical Unit

Instead of having individual registers performing the micro-operations, computer system provides a number of registers connected to a common unit called as Arithmetic Logical Unit (ALU). ALU is the main and one of the most important unit inisde CPU of computer. All the logical and mathematical operations of computer are performed here. The contents of specific register is placed in the in the input of ALU. ALU performs the given operation and then transfer it to the destination register.



CHAPTER SIX

ADDRESSING MODES AND INSTRUCTION CYCLE

6.1 ADDRESSING MODES:- The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The purpose of using addressing modes is as follows:

1. To give the programming versatility to the user.
2. To reduce the number of bits in addressing field of instruction.

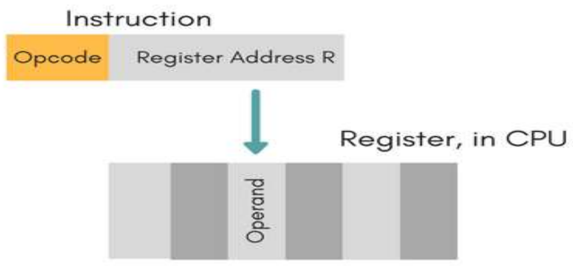
Types of Addressing Modes

Different types of addressing modes:

1**) Immediate Mode**:- In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

2) Register Mode:- In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.



Advantages

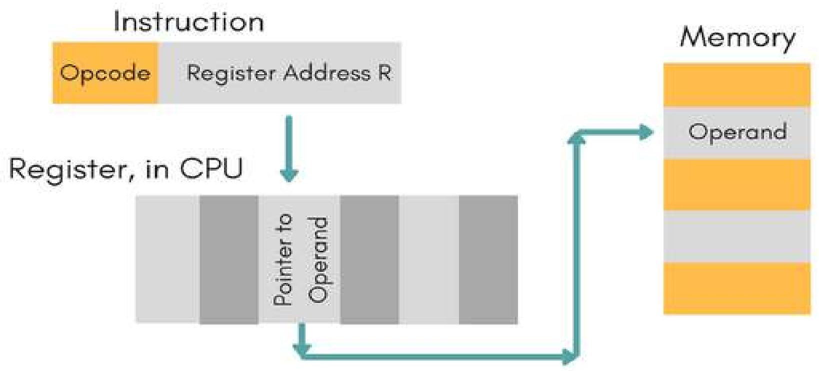
- Shorter instructions and faster instruction fetch.
- Faster memory access to the operand(s)

Disadvantages

- Very limited address space
- Using multiple registers helps performance but it complicates the instructions.

Register Indirect Mode

In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.



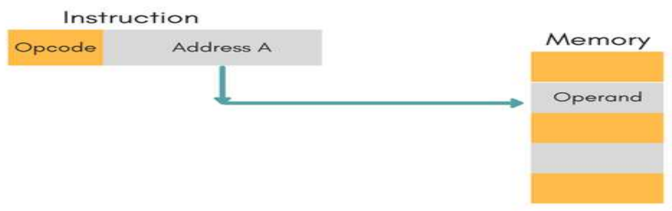
Auto Increment/Decrement Mode

In this the register is incremented or decremented after or before its value is used.

Direct Addressing Mode

In this mode, effective address of operand is present in instruction itself.

- Single memory reference to access data.
- No additional calculations to find the effective address of the operand.

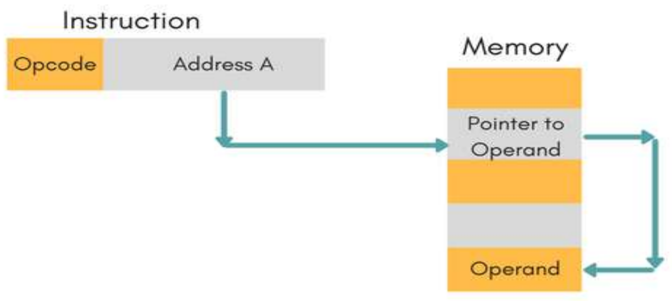


For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

NOTE: Effective Address is the location where operand is present.

Indirect Addressing Mode

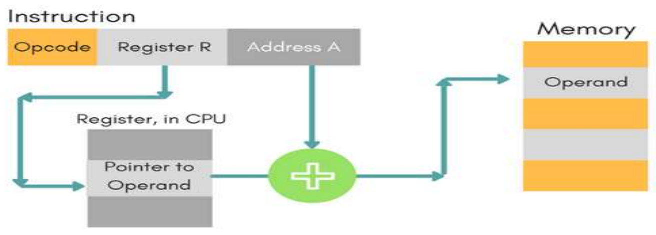
In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.



Displacement Addressing Mode

In this the contents of the indexed register is added to the Address part of the instruction, to obtain the effective address of operand.

EA = A + (R), In this the address field holds two values, A(which is the base value) and R(that holds the displacement), or vice versa.



Relative Addressing Mode

In this the contents of PC (Program Counter) is added to address part of instruction to obtain the effective address.

EA = A + (PC), where EA is effective address and PC is program counter.

The operand is A cells away from the current cell (the one pointed to by PC)

Base Register Addressing Mode

It is again a version of Displacement addressing mode. This can be defined as EA = A + (R), where A is displacement and R holds pointer to base address.

Stack Addressing Mode

In this mode, operand is at the top of the stack. For example: ADD, this instruction will *POP* top two items from the stack, add them, and will then *PUSH* the result to the top of the stack.

6.2 INSTRUCTION CYCLE: - An instruction cycle, also known as **fetch-decode-execute cycle** is the basic operational process of a computer. This process is repeated continuously by CPU from boot up to shut down of computer.

Following are the steps that occur during an instruction cycle:

1. Fetch the Instruction

The instruction is fetched from memory address that is stored in PC(Program Counter) and stored in the instruction register IR. At the end of the fetch operation, PC is incremented by 1 and it then points to the next instruction to be executed.

2. Decode the Instruction

The instruction in the IR is executed by the decoder.

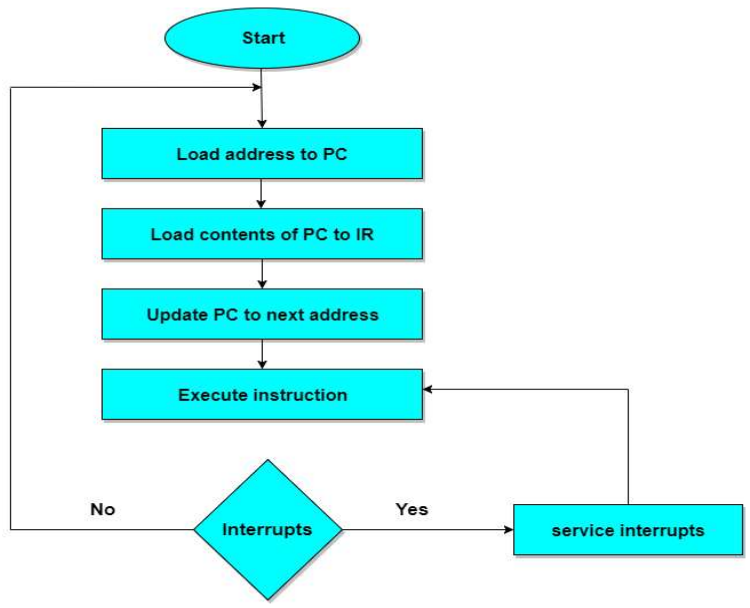
3. Read the Effective Address

If the instruction has an indirect address, the effective address is read from the memory. Otherwise operands are directly read in case of immediate operand instruction.

4. Execute the Instruction

The Control Unit passes the information in the form of control signals to the functional unit of CPU. The result generated is stored in main memory or sent to an output device.

The cycle is then repeated by fetching the next instruction. Thus in this way the instruction cycle is repeated continuously.



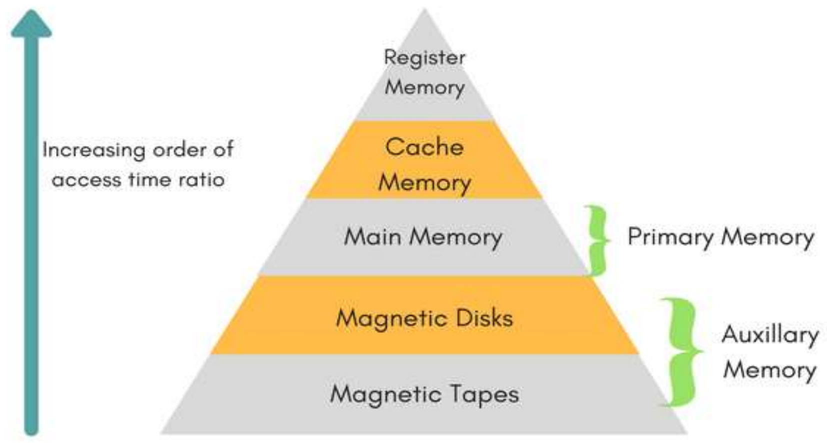
Instruction cycle (fetch-decode-execute cycle)

6.3 MEMORY ORGANIZATION IN COMPUTER ARCHITECTURE

A memory unit is the collection of storage units or devices together. The memory unit stores the binary information in the form of bits. Generally, memory/storage is classified into 2 categories:

- **Volatile Memory**: This loses its data, when power is switched off.
- **Non-Volatile Memory**: This is a permanent storage and does not lose any data when power is switched off.

Memory Hierarchy



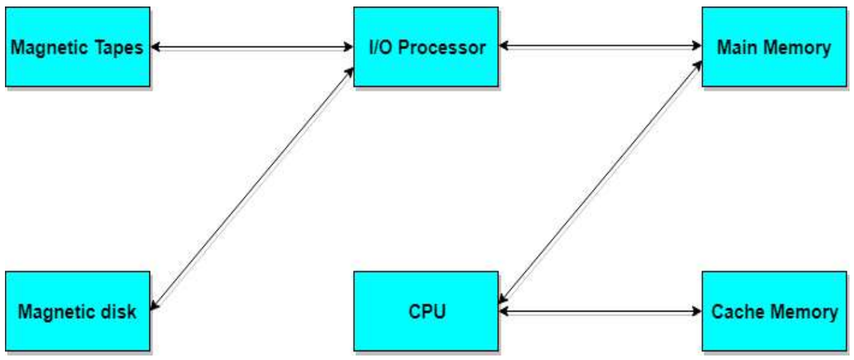
The total memory capacity of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the slow Auxiliary Memory to fast Main Memory and to smaller Cache memory.

Auxillary memory access time is generally **1000 times** that of the main memory, hence it is at the bottom of the hierarchy.

The **main memory** occupies the central position because it is equipped to communicate directly with the CPU and with auxiliary memory devices through Input/output processor (I/O).

When the program not residing in main memory is needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use.

The **cache memory** is used to store program data which is currently being executed in the CPU. Approximate access time ratio between cache memory and main memory is about **1 to 7~10**



Memory Access Methods

Each memory type is a collection of numerous memory locations. To access data from any memory, first, it must be located and then the data is read from the memory location. Following are the methods to access information from memory locations:

1. **Random Access**: Main memories are random access memories, in which each memory location has a unique address. Using this unique address any memory location can be reached in the same amount of time in any order.
2. **Sequential Access**: This method allows memory access in a sequence or in order.
3. **Direct Access**: In this mode, information is stored in tracks, with each track having a separate read/write head.

Main Memory

The memory unit that communicates directly within the CPU, Auxillary memory and Cache memory, is called main memory. It is the central storage unit of the computer system. It is a large and fast memory used to store data during computer operations. Main memory is made up of **RAM** and **ROM**, with RAM integrated circuit chips holing the major share.

- RAM: Random Access Memory
  - **DRAM**: Dynamic RAM, is made of capacitors and transistors, and must be refreshed every 10~100 ms. It is slower and cheaper than SRAM.
  - **SRAM**: Static RAM, has a six transistor circuit in each cell and retains data, until powered off.
  - **NVRAM**: Non-Volatile RAM, retains its data, even when turned off. Example: Flash memory.
- ROM: Read Only Memory, is non-volatile and is more like a permanent storage for information. It also stores the **bootstrap loader** program, to load and start the operating system when computer is turned on. **PROM**(Programmable ROM), **EPROM**(Erasable PROM) and **EEPROM**(Electrically Erasable PROM) are some commonly used ROMs.

Auxiliary Memory:- Devices that provide backup storage are called auxiliary memory. **For example:** Magnetic disks and tapes are commonly used auxiliary devices. Other devices used as auxiliary memory are magnetic drums, magnetic bubble memory and optical disks.

It is not directly accessible to the CPU, and is accessed using the Input/Output channels.

Cache Memory:- The data or contents of the main memory that are used again and again by CPU, are stored in the cache memory so that we can easily access that data in 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 onto the main memory. It also transfers block of recent data into the cache and keeps on deleting the old data in cache to accommodate the new one.

Hit Ratio:- The performance of cache memory is 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 then it counts as a **miss**.

The ratio of the number of hits to the total CPU references to memory is called hit ratio.

Hit Ratio = Hit/(Hit + Miss)

Associative Memory:- It is also known as **content addressable memory (CAM)**. It is a memory chip in which each bit position can be compared. In this the content is compared in each bit cell which allows very fast table lookup. Since the entire chip can be compared, contents are randomly stored without considering addressing scheme. These chips have less storage capacity than regular memory chips.

CHAPTER SEVEN

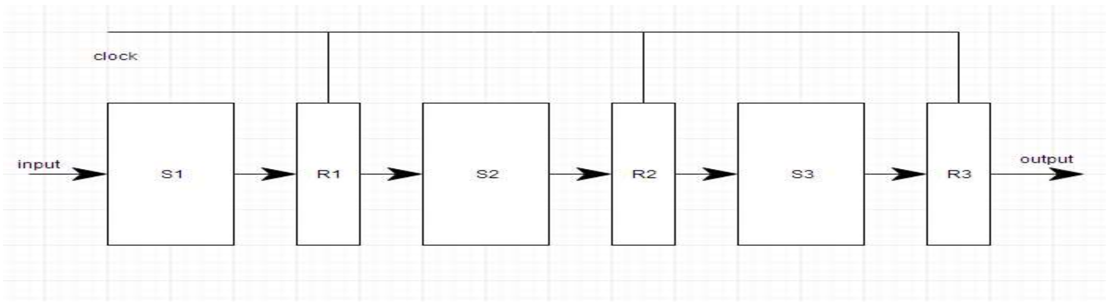
PIPELINING

7.1 PIPELINING:- Pipelining is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process. It is also known as **pipeline processing**.

Pipelining is a technique where multiple instructions are overlapped during execution. Pipeline is divided into stages and these stages are connected with one another to form a pipe like structure. Instructions enter from one end and exit from another end.

Pipelining increases the overall instruction throughput.

In pipeline system, each segment consists of an input register followed by a combinational circuit. The register is used to hold data and combinational circuit performs operations on it. The output of combinational circuit is applied to the input register of the next segment.



Pipeline system is like the modern day assembly line setup in factories. For example in a car manufacturing industry, huge assembly lines are setup and at each point, there are robotic arms to perform a certain task, and then the car moves on ahead to the next arm.

7.2 TYPES OF PIPELINE

It is divided into 2 categories: (1) Arithmetic Pipeline (2) Instruction Pipeline

1. **Arithmetic Pipeline:-** Arithmetic pipelines are usually found in most of the computers. They are used for floating point operations, multiplication of fixed point numbers etc. For example: The input to the Floating Point Adder pipeline is:

*X = A*2^a*

*Y = B*2^b*

Here A and B are mantissas (significant digit of floating point numbers), while **a** and **b** are exponents.

The floating point addition and subtraction is done in 4 parts:

1. Compare the exponents.
2. Align the mantissas.
3. Add or subtract mantissas
4. Produce the result.

Registers are used for storing the intermediate results between the above operations.

(2) Instruction pipeline:- In this a stream of instructions can be executed by overlapping *fetch*, *decode* and *execute* phases of an instruction cycle. This type of technique is used to increase the throughput of the computer system.

An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously. The pipeline will be more efficient if the instruction cycle is divided into segments of equal duration.

7.2 PIPELINE CONFLICTS:- There are some factors that cause the pipeline to deviate its normal performance. Some of these factors are given below:

1. Timing Variations:- All stages cannot take same amount of time. This problem generally occurs in instruction processing where different instructions have different operand requirements and thus different processing time.

2. Data Hazards:- When several instructions are in partial execution, and if they reference same data then the problem arises. We must ensure that next instruction does not attempt to access data before the current instruction, because this will lead to incorrect results.

3. Branching:- In order to fetch and execute the next instruction, we must know what that instruction is. If the present instruction is a conditional branch, and its result will lead us to the next instruction, then the next instruction may not be known until the current one is processed.

4. Interrupts:- Interrupts set unwanted instruction into the instruction stream. Interrupts affect the execution of instruction.

5. Data Dependency:- It arises when an instruction depends upon the result of a previous instruction but this result is not yet available.

7.3 ADVANTAGES OF PIPELINING

1. The cycle time of the processor is reduced.
2. It increases the throughput of the system
3. It makes the system reliable.

7.4 DISADVANTAGES OF PIPELINING

1. The design of pipelined processor is complex and costly to manufacture.
2. The instruction latency is more.

CHAPTER EIGHT

MODES OF INPUT/OUTPUT DATA TRANSFER

8.1 DATA TRANSFER:- Data transfer between the central unit and I/O devices can be handled in generally three types of modes which are given below:

1. Programmed I/O
2. Interrupt Initiated I/O
3. Direct Memory Access

(1) Programmed I/O:- Programmed I/O instructions are the result of I/O instructions written in computer program. Each data item transfer is initiated by the instruction in the program.

Usually the program controls data transfer to and from CPU and peripheral. Transferring data under programmed I/O requires constant monitoring of the peripherals by the CPU.

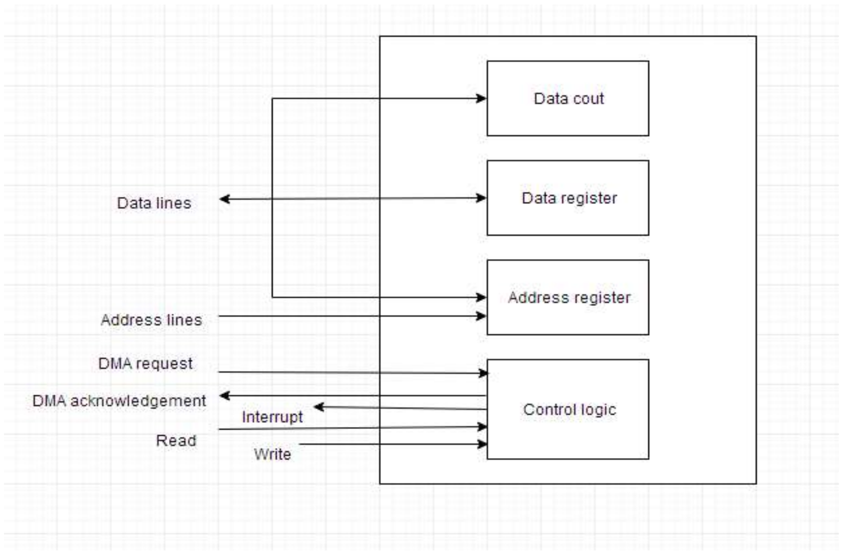
(**2) Interrupt Initiated I/O:-** In the programmed I/O method the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy needlessly.

This problem can be overcome by using **interrupt initiated I/O**. In this when the interface determines that the peripheral is ready for data transfer, it generates an interrupt. After receiving the interrupt signal, the CPU stops the task which it is processing and service the I/O transfer and then returns back to its previous processing task.

(3) Direct Memory Access:- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This technique is known as **DMA**.

In this, the interface transfer data to and from the memory through memory bus. A DMA controller manages to transfer data between peripherals and memory unit.

Many hardware systems use DMA such as disk drive controllers, graphic cards, network cards and sound cards etc. It is also used for intra chip data transfer in multicore processors. In DMA, CPU would initiate the transfer, do other operations while the transfer is in progress and receive an interrupt from the DMA controller when the transfer has been completed.



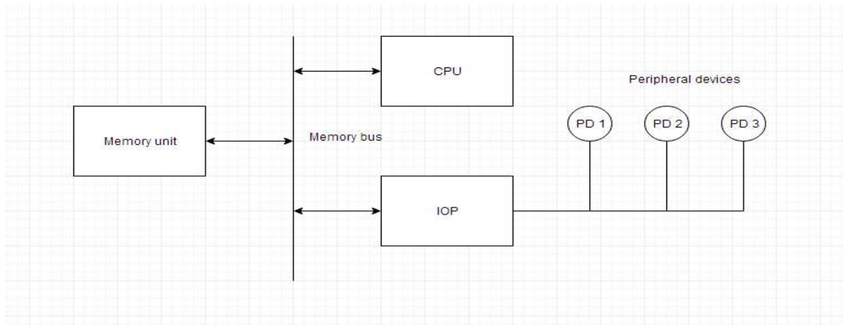
Block diagram of DMA

8.2 INPUT/OUTPUT PROCESSOR:- An input-output processor (IOP) is a processor with direct memory access capability. In this, the computer system is divided into a memory unit and number of processors. Each IOP controls and manage the input-output tasks. The IOP is similar to CPU except that it handles only the details of I/O processing. The IOP can fetch and execute its own instructions. These IOP instructions are designed to manage I/O transfers only.

8.3 BLOCK DIAGRAM OF I/O PROCESSOR

Below is a block diagram of a computer along with various I/O Processors. The memory unit occupies the central position and can communicate with each processor.

The CPU processes the data required for solving the computational tasks. The IOP provides a path for transfer of data between peripherals and memory. The CPU assigns the task of initiating the I/O program. The IOP operates independent from CPU and transfer data between peripherals and memory.



The communication between the IOP and the devices is similar to the program control method of transfer. And the communication with the memory is similar to the direct memory access method.

In large scale computers, each processor is independent of other processors and any processor can initiate the operation.

The CPU can act as master and the IOP act as slave processor. The CPU assigns the task of initiating operations but it is the IOP, who executes the instructions, and not the CPU. CPU instructions provide operations to start an I/O transfer. The IOP asks for CPU through interrupt.

Instructions that are read from memory by an IOP are also called *commands* to distinguish them from instructions that are read by CPU. Commands are prepared by programmers and are stored in memory. Command words make the program for IOP. CPU informs the IOP where to find the commands in memory.

CHAPTER NINE

INTERRUPTS AND MODES OF TRANSMISSION

9.1 INTERRUPT:- Data transfer between the CPU and the peripherals is initiated by the CPU. But the CPU cannot start the transfer unless the peripheral is ready to communicate with the CPU. When a device is ready to communicate with the CPU, it generates an interrupt signal. A number of input-output devices are attached to the computer and each device is able to generate an interrupt request.

The main job of the interrupt system is to identify the source of the interrupt. There is also a possibility that several devices will request simultaneously for CPU communication. Then, the interrupt system has to decide which device is to be serviced first.

9.1.1 Priority Interrupt:- A priority interrupt is a system which decides the priority at which various devices, which generates the interrupt signal at the same time, will be serviced by the CPU. The system has authority to decide which conditions are allowed to interrupt the CPU, while some other interrupt is being serviced. Generally, devices with high speed transfer such as *magnetic disks* are given high priority and slow devices such as *keyboards* are given low priority.

When two or more devices interrupt the computer simultaneously, the computer services the device with the higher priority first.

9.2 TYPES OF INTERRUPTS: Following are some different types of interrupts:

9.2.1 Hardware Interrupts:- When the signal for the processor is from an external device or hardware then this interrupts is known as **hardware interrupt**.

Let us consider an example: when we press any key on our keyboard to do some action, then this pressing of the key will generate an interrupt signal for the processor to perform certain action. Such an interrupt can be of two types:

1. **Maskable Interrupt:-** The hardware interrupts which can be delayed when a much high priority interrupt has occurred at the same time.

2. **Non Maskable Interrupt**:- The hardware interrupts which cannot be delayed and should be processed by the processor immediately.

9.2.2 Software Interrupts:- The interrupt that is caused by any internal system of the computer system is known as a **software interrupt**. It can also be of two types:

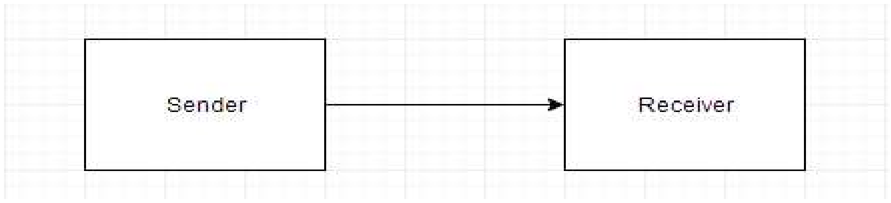
1. **Normal Interrupt**:- The interrupts that are caused by software instructions are called **normal software interrupts**.

2. **Exception**:- Unplanned interrupts which are produced during the execution of some program are called **exceptions**, such as division by zero.

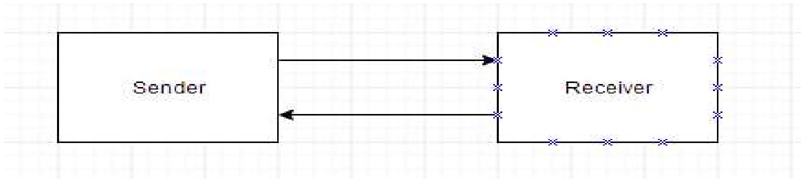
9.3 MODES OF TRANSMISSION

Data can be transmitted between 2 points by three different modes:

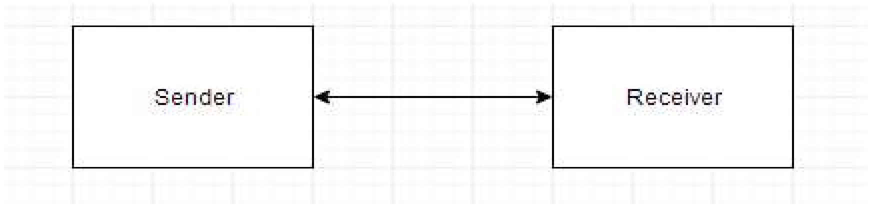
9.3.1 Simplex:- A simplex line carries information in one direction only. In this mode receiver cannot communicate with the sender to indicate the occurrence of errors that means only sender can send data but receiver cannot. **For example:** Radio and Television Broadcasting.



9.3.2 Half Duplex:- In half duplex mode, system is capable of transmitting data in both directions but data can be transmitted in one direction only at a time. A pair of wires is needed for this mode. For example: **Walkie - Talkie.**



9.3.3 Full Duplex:- In this mode data can be send and received in both directions simultaneously. In these four wire link is used. For example: Video Calling, Audio calling etc.



CHAPTER TEN

PROTOCOL AND COMMON CPU COMPONENTS

10.1 PROTOCOL:- A **Protocol** is a set of rules that are followed by interconnecting devices to ensure that all data is passed correctly without any error. The orderly transmission of data in a data link can be accomplished by a protocol.

10.1.1 Types of Protocols

There are two types of protocols:

1. **Character Oriented Protocol**:- It is based on the binary code of character set. The code is mostly used in ASCII. It includes upper case and lower case letters, numerals and variety of special symbols. The characters that control the transmission is called communication control characters.

2. **Bit Oriented Protocol:-** It does not use characters in its control field and is independent of any code. It allows the transmission of serial bit stream of any length without the implication of character boundaries**.**

10.2 COMMON CPU COMPONENTS

The central processing unit (CPU) consists of six main components: Control Unit (CU), Arithmetic Logic Unit (ALU), Registers, Cache, Buses, Clock

All the components work together to allow processing and system control.

1. Control unit (CU):- The CU provides several functions:

- It fetches, decodes and executes instructions
- It issues control signals that control hardware components within the CPU
- It transfers data and instructions around the system

2. **Arithmetic Logic Unit (ALU):-** The ALU has two main functions:

- It performs arithmetic and logical operations (decisions).
- It acts as a gateway between primary storage and secondary storage - data transferred between them passes through the ALU.

3. **Registers:-** Registers are small amounts of high-speed memory contained within the CPU. They are used by the processor to store small amounts of data that are needed during processing, such as:

- the address of the next instruction to be executed
- the current instruction being decoded
- the results of calculations

Different processors have different numbers of registers for different purposes. Most have some, or all, of the following: Program Counter (PC), Memory Address Register (MAR), Memory Data Register (MDR), Current Instruction Register (CIR), Accumulator (ACC)

4. **Cache:-** Cache is a small amount of high-speed random access memory (RAM) built directly within the processor. It is used to temporarily hold data and instructions that the processor is likely to reuse. This allows for faster processing, as the processor does not have to wait for the data and instructions to be fetched from the RAM.

5. **Clock:-** The CPU contains a clock which, along with the CU, is used to coordinate all of the computer's components. The clock sends out a regular electrical pulse which synchronises (keeps in time) all the components.

The frequency of the pulses is known as clock speed. Clock speed is measured in hertz (Hz). The greater the speed, the more instructions can be performed in any given moment of time.

6. **Buses:-** A bus is a high-speed internal connection. Buses are used to send control signals and data between the processor and other components.

Three types of bus are used.

- **Address bus** - carries memory addresses from the processor to other components such as primary storage and input/output devices. The address bus is unidirectional.
- **Data bus** - carries the data between the processor and other components. The data bus is bidirectional.
- **Control bus** - carries control signals from the processor to other components. The control bus also carries the clock's pulses. The control bus is unidirectional.