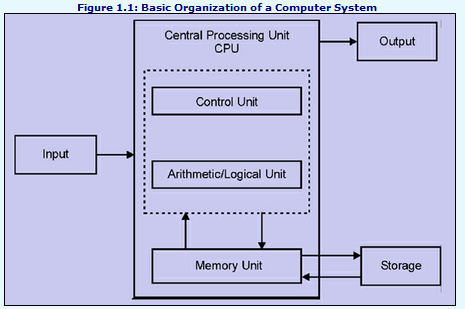
# Module - 1

**Syllabus: -**

* Basics of Computer Organization: Basic organization of the stored program computer and operation sequence for execution of a program, Von Neumann & Harvard Architecture.
* RISC vs. CISC based architecture.
* Fetch, decode and execute cycle
* Concept of registers and storage, Instruction format, Instruction sets and addressing modes.
* Basics of Control Unit Design - hardwired and micro programmed control, Horizontal and Vertical micro instruction.

**Basic organization of the stored program computer**



**Central Processing Unit (CPU)**: This is the brain of the computer where actual computation takes place. It consists of the Arithmetic Logic Unit (ALU) and the Control Unit.

**Arithmetic Logic Unit (ALU)**: The ALU performs arithmetic (addition, subtraction, etc.) and logical (AND, OR, NOT, etc.) operations on data.

**Control Unit**: The control unit fetches instructions from the memory, decodes them and controls the operations of the CPU. It directs and coordinates the operations of all other parts of the computer. It directs all input and output flows, fetches code for instructions and controls the data movement around the system.

**Registers & Buses**: Registers are the small and fast storage locations within the CPU used to store data temporarily during processing. Buses are sets of parallel wires connecting various components of the computer, allowing them to communicate with each other.

**Memory Unit**: This includes RAM (Random Access Memory) and ROM (Read-Only Memory). RAM is a volatile memory used to store data and program instructions temporarily during execution. ROM is a non-volatile memory that stores firmware and bootstrap programs.

**Input/Output (I/O) Devices:** These include devices such as keyboards, mice, monitors, printers, etc., which allow users to interact with the computer and receive output.

**Operation sequence for the execution of a program**

**Fetch:** The CPU fetches the next instruction from the memory. The address of the next instruction is typically held in a special register called as the program counter (PC).

**Decode:** The fetched instruction is decoded to determine what operation needs to be performed and what operands are involved.

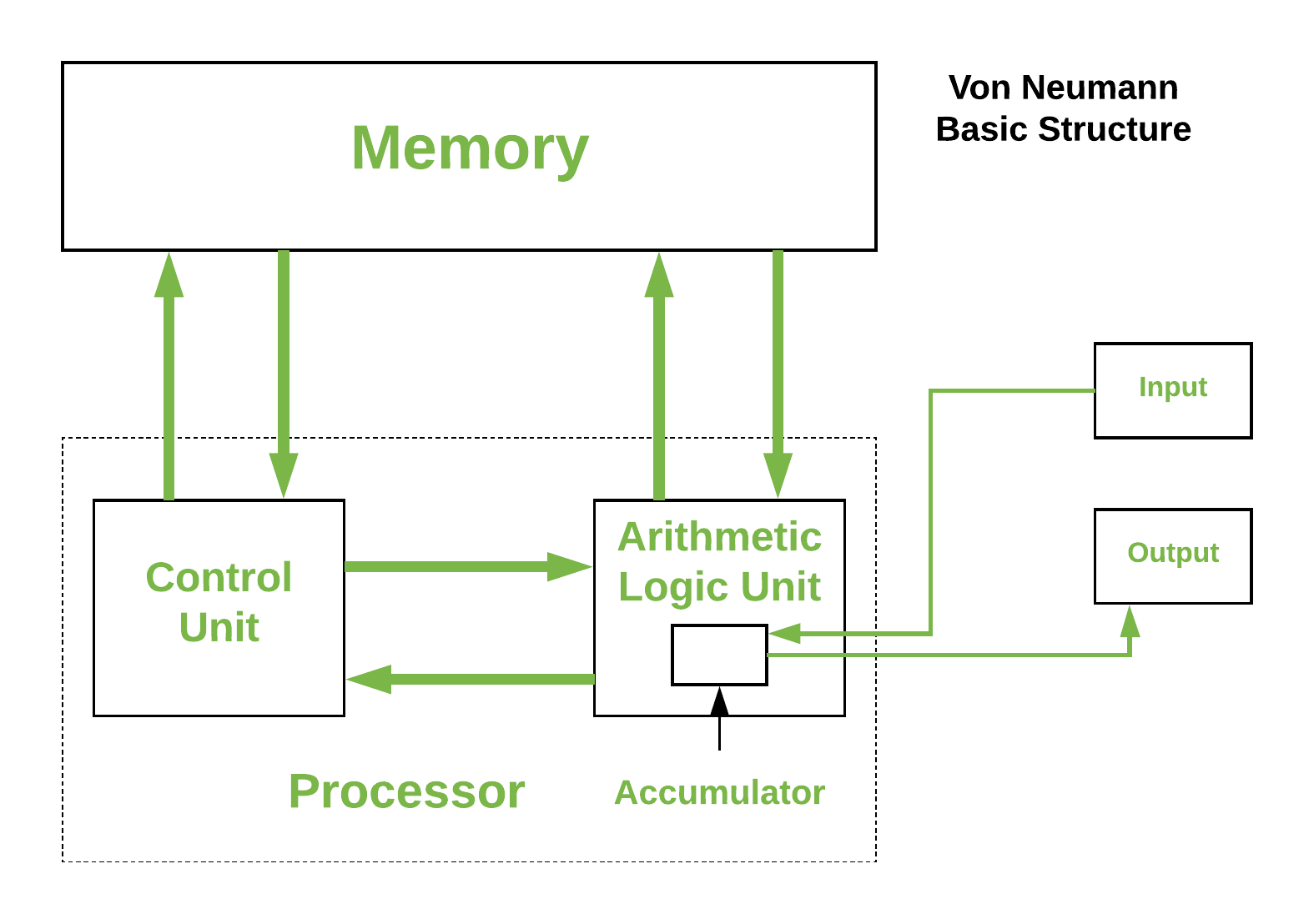
**Execute:** The CPU executes the instruction by performing the specified operation on the data. This may involve calculations, data movement, logical operations, etc.

**Store:** If the instruction produces a result that needs to be stored, such as the result of a calculation or the updated value of a variable, it is stored back in memory or in a register.

**Update Program Counter:** After completing the execution of an instruction, the program counter (PC) is typically incremented to point to the next instruction in memory, so that the cycle can repeat for the next instruction.

**Von Neumann Architecture**

Modern computers are based on a stored-program concept introduced by John Von Neumann. In this stored-program concept, programs and data are stored in a separate storage unit called memories and are treated the same.



It is also known as **ISA**(Instruction Set Architecture) Computer and is having three basic units:-

1. The Central Processing Unit (CPU)
2. The Main Memory Unit
3. The Input/Output Device

**1. Central Processing Unit**

**The central processing unit is defined as an electric circuit used for executing the instruction in the computer program.**

**It has following major components:**

1. **Control Unit(CU)**

**B. Arithmetic and Logic Unit(ALU)**

**C. Variety of Registers**

**Control Unit**   
 A control unit (CU) directs and coordinates the operation of all other parts of the computer. It directs all input and output flow, fetches code for instructions, and controls the data movement around the system.

**Arithmetic and Logic Unit (ALU)**   
 The arithmetic logic unit is that part of the CPU that handles all the calculations of the CPU like Addition, Subtraction, Comparisons. It performs Logical Operations, Bit Shifting Operations, and Arithmetic operations.

**Registers**

Registers refer to high-speed storage areas in the CPU. The data processed by the CPU are fetched from the registers. There are different types of registers used in architecture:

1. **Accumulator:** It stores the results of calculations made by ALU. It holds the intermediate of arithmetic and logical operations.It act as  a temporary storage location or device.
2. **Program Counter (PC):** Keeps track of the memory location of the next instructions so that it can send the address of the memory location to the Memory Address Register (MAR).

1. **Memory Address Register (MAR):** It stores the memory locations of instructions that need to be fetched from the memory or store in the memory.

1. **Memory Data Register (MDR):** It stores the instructions fetched from memory or any data that needs to be transferred to or stored in the memory.

1. **Current Instruction Register (CIR):** It stores the most recently fetched instructions while it is waiting to be coded and executed.
2. **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.

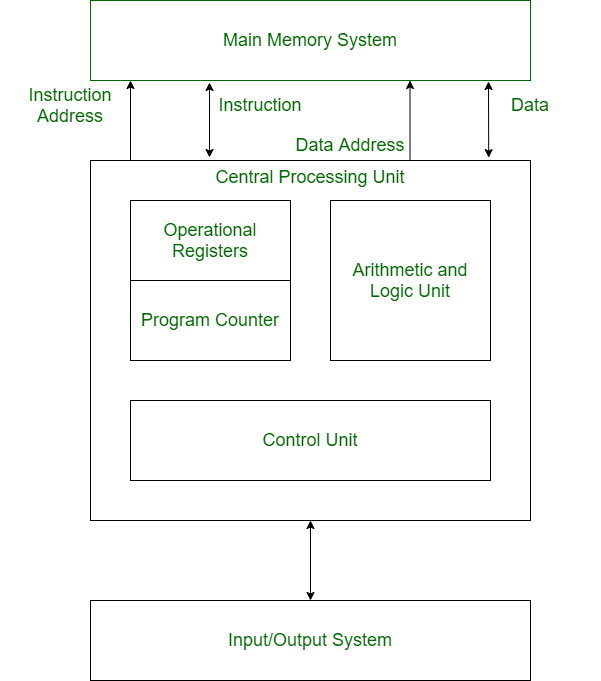
Note: -

**“Von Neumann bottleneck”**  
Whatever we do to enhance performance, we cannot get away from the fact that instructions can only be done one at a time and can only be carried out sequentially. Both of these factors hold back the competence of the CPU. This is commonly referred to as the ‘Von Neumann bottleneck’. We can provide a Von Neumann processor with more cache, more RAM, or faster components but if original gains are to be made in CPU performance then an influential inspection needs to take place of CPU configuration.

This architecture is very important and is used in our PCs and even in Super Computers.

**Harvard Architecture**

**Harvard Architecture** is the computer architecture that contains separate storage and separate buses (signal path) for instruction and data. It was basically developed to overcome the bottleneck of Von Neumann’s Architecture. The main advantage of having separate buses for instruction and data is that the CPU can access instructions and read/write data at the same time.



**Buses**

Buses are used as signal pathways. In Harvard architecture, there are separate buses for both instruction and data. Types of Buses:

1. **Data Bus:** It carries data among the main memory, processor, and I/O devices.
2. **Data Address Bus:** It carries the address of data from the processor to the main memory.
3. **Instruction Bus:** It carries instructions among the main memory, processor, and I/O devices.
4. **Instruction Address Bus:** It carries the address of instructions from the processor to the main memory system.

**Operational Registers**

There are different types of registers involved in it which are used for storing addresses of different types of instructions. *For example*, the Memory Address Register and Memory Data Register are operational registers.

1. **Program Counter:** It has the location of the next instruction to be executed. The program counter then passes this next address to the memory address register.
2. **Arithmetic and Logic Unit:** The arithmetic logic unit is part of the CPU that operates all the calculations needed. It performs addition, subtraction, comparison, logical Operations, bit Shifting Operations, and various arithmetic operations.
3. **Control Unit:**  The control unit fetches instructions from the memory, decodes them and controls the operations of the CPU. It directs and coordinates the operations of all other parts of the computer. It directs all input and output flows, fetches code for instructions and controls the data movement around the system.
4. **Input/Output System:** Input devices are used to read data into main memory with the help of CPU input instruction. The information from a computer as output is given through Output devices. The computer gives the results of computation with the help of output devices.

**Features:**

1. **Separate memory spaces**
2. **Fixed instruction length**
3. **Parallel instruction and data access**
4. **More efficient memory usage**
5. **Suitable for embedded systems**
6. **Limited flexibility**

****Reduced Instruction Set Computer (RISC) Architecture****

**It is a microprocessor architecture that uses a small, highly-optimized set of instructions instead of the specialized set of instructions found in other architectures.**

### ****Characteristics of RISC****

1. Simple instruction decoding.
2. Instruction comes under size of one word.
3. Instruction takes a single clock cycle to get executed.
4. More general-purpose registers.
5. Simple Addressing Modes.
6. Fewer Data types.
7. A pipeline can be achieved.

### Advantages of RISC

1. **Simple instructions**
2. **Faster execution**
3. **Low power consumption**

### Disadvantages of RISC

1. **More instructions required**
2. **Increased memory usage**
3. **Higher cost**

****Complex Instruction Set Computer (CISC) Architecture****

It's a computer architecture that uses a large number of instructions, some of which operate with the data in the registers and others directly operate with the data in the system's memory.

### ****Characteristics of CISC****

1. Complex instruction decoding.
2. Instructions are larger than one-word size.
3. Instruction may take more than a single clock cycle to get executed.
4. Less number of general-purpose registers
5. Complex Addressing Modes.
6. More Data types.

### Advantages of CISC

1. **Reduced code size**
2. **More memory efficiency**
3. **Widely used**

### Disadvantages of CISC

1. **Slower execution**
2. **More complex design**
3. **High power consumption**

****Addressing Modes****

The term addressing modes refers to a way in which the operand of an instruction is specified.

Types of Addressing modes: -

1. **Implied addressing mode -->** In implied addressing mode the operand is specified in the instruction itself. In this mode the data is either 8 bits or 16 bits long and data is the part of instruction itself.

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

1. **Immediate addressing mode -->** Immediate addressing mode is a way of addressing in which the data is stored in the address field of the instruction. Limitation is that, the range of the constant is restricted buy the size of the address field.

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

1. **Register addressing mode -->** In register addressing mode the operand is stored in one of the 8bit or 16bit general purpose registers and that register is specified in the instruction to access the data.

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

1. **Register Indirect addressing mode -->** In register indirect addressing mode the operand’s offset address is stored in one of the following registers (BP, BX, SI, DI) as specified in the instruction and that register will contain the effective address of the data from which the data can be accessed.

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

addressed by the register BX to the register AX)

1. **Direct/Absolute addressing mode -->** In this addressing mode the operand’s offset address is given in the instruction as an 8bit or 16bit displacement element. In this addressing mode the 16bit effective address of the data is the part of the instruction itself.

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

1. **Indirect addressing mode -->** In this addressing mode the address field of the instruction contains the address of the effective address, here 2 references are required: - 1st reference to get the effective address, 2nd reference to access the data.
2. **Indexed addressing mode --> I**n this addressing mode the operand’s offset address is the sum of the content of any one of the index registers(SI or DI) and an 8bit or 16bit displacement.

Example: MOV AX, [SI +05]

1. **Based Indexed Addressing: in this addressing mode, t**he operand’s offset address is the sum of the content of a base register BX or BP and an index register SI or DI.

Example: ADD AX, [BX+SI]

****Instruction Format****

### **1. Addressing Mode**

The data is represented in the instruction format with the help of addressing mode

The addressing mode is the first part of the instruction format

The data can either be stored in the memory of a computer or it can be located in the register of the CPU

### **2. Operation Code( OPCODE)**

The operation code gives instructions to the processor to perform the specific Operation

The operation code is the second part of the instruction format

### **3. OPERAND**

It is the part of the instruction format that specifies the data or the address of the data

Depending upon the processor of the computer the instruction format contains zero to three operands

**Types Of Instruction Format: -**

### **1. Zero(0) Address Instruction format**

The instruction format in which there is no address field is called zero address instruction.

In zero address instruction format, stacks are used.

In zero order instruction format, there is no operand.

### **2. One(1) Address Instruction format**

The instruction format in which the instruction uses only one address field is called the one address instruction format

In this type of instruction format, one operand is in the accumulator and the other is in the memory location

It has only one operand

It has two special instructions LOAD and STORE

### **3. Two(2) Address Instruction format**

The instruction format in which the instruction uses only two address fields is called the two address instruction format

This type of instruction format is the most commonly used instruction format

As in one address instruction format, the result is stored in the accumulator only, but in two addresses instruction format the result can be stored in different locations

This type of instruction format has two operands

It requires shorter assembly language instructions

### **4. Three(3) Address Instruction format**

The instruction format in which the instruction uses the three address fields is called the three address instruction format

It has three operands

It requires shorter assembly language instructions

It requires more bits

**Control Unit**

The control unit fetches instructions from the memory, decodes them and controls the operations of the CPU. It directs and coordinates the operations of all other parts of the computer. It directs all input and output flows, fetches code for instructions and controls the data movement around the system.

There are two types of control unit: -

1. Hardwired Control Unit
2. Micro-programmed control Unit

Hardwired Control Unit

It is a type of control unit used in a computer architecture to manage the execution of an instruction in a processor. It uses dedicated hardwired circuitry to decode and execute the instructions directly. The control signals required for executing the instructions are generated by combinational logic circuits which typically implemented by using logic gates such as AND, OR, NOT. These circuits are designed to generate the necessary control signals based on the instruction opcode and other information.

Advantages : -

1. Speed
2. Simplicity
3. Parallelism
4. Reliability

Disadvantages: -

1. Lack of flexibility
2. Complexity for complex instruction sets
3. Difficulty in debugging and testing
4. Limited adaptability

Micro programmed Control Unit

It is a type of control unit used in a computer architecture to manage the execution of an instruction in a processor. It uses dedicated hardwired circuits to generate the control signals, micro programmed control unit uses microcode stored in the memory to interpret and execute the instructions. The control signals required for executing the instructions generated by executing sequences of microinstruction stored in a control memory.

Advantages : -

1. Flexibility
2. Simplicity of design
3. Ease of debugging and testing

Disadvantages: -

1. Slower execution
2. Complexity of micro code