Digital Computer Organization

- The digital computer is a digital system that performs various computational tasks.
- The word digital implies that the information in the computer is represented by variables that take a limited number of discrete values. These values are processed internally by components that can maintain a limited number of discrete states.
- The decimal digits 0, 1, 2, ..., 9, for example, provide 10 discrete values. The first electronic digital computer, developed in the late 1940s, was used primarily for numerical computations and the discrete elements were the digits. From this application the term digital computer emerged.

- In practice, digital computers function more reliably if only two states are used. Because of the physical restriction of components, and because human logic tends to be binary (i.e. true or false, yes or no statements), digital components that are constrained to take discrete values are further constrained to take only two values and are said to be binary.

- Digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a bit. Information is represented in digital computers in groups of bits. By using various coding techniques, groups of bits can be made to represent not only binary numbers but also other discrete symbols, such as decimal digits or letters of the alphabet.

Digital Computers: Computer Organization

- Computer Organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system.

- The various components are assumed to be in place and the task is to investigate the organizational structure to verify that the computer parts operate as intended.

Digital Computers: Computer Design

- Computer Design is concerned with the hardware design of the computer. Once the computer specifications are formulated, it is the task of the designer to develop hardware for the system.

- Computer design is concerned with the determination of what hardware should be used and how the parts should be connected. This aspect of computer hardware is sometimes referred to as computer implementation.

Digital Computers: Computer Architecture

Computer Architecture is concerned with the structure and behavior of the computer as seen by the user.

It includes the information, formats, the instruction set, and techniques for addressing memory.

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.

Two basic types of computer architecture are:

1. **von Neumann architecture**
2. **Harvard architecture**

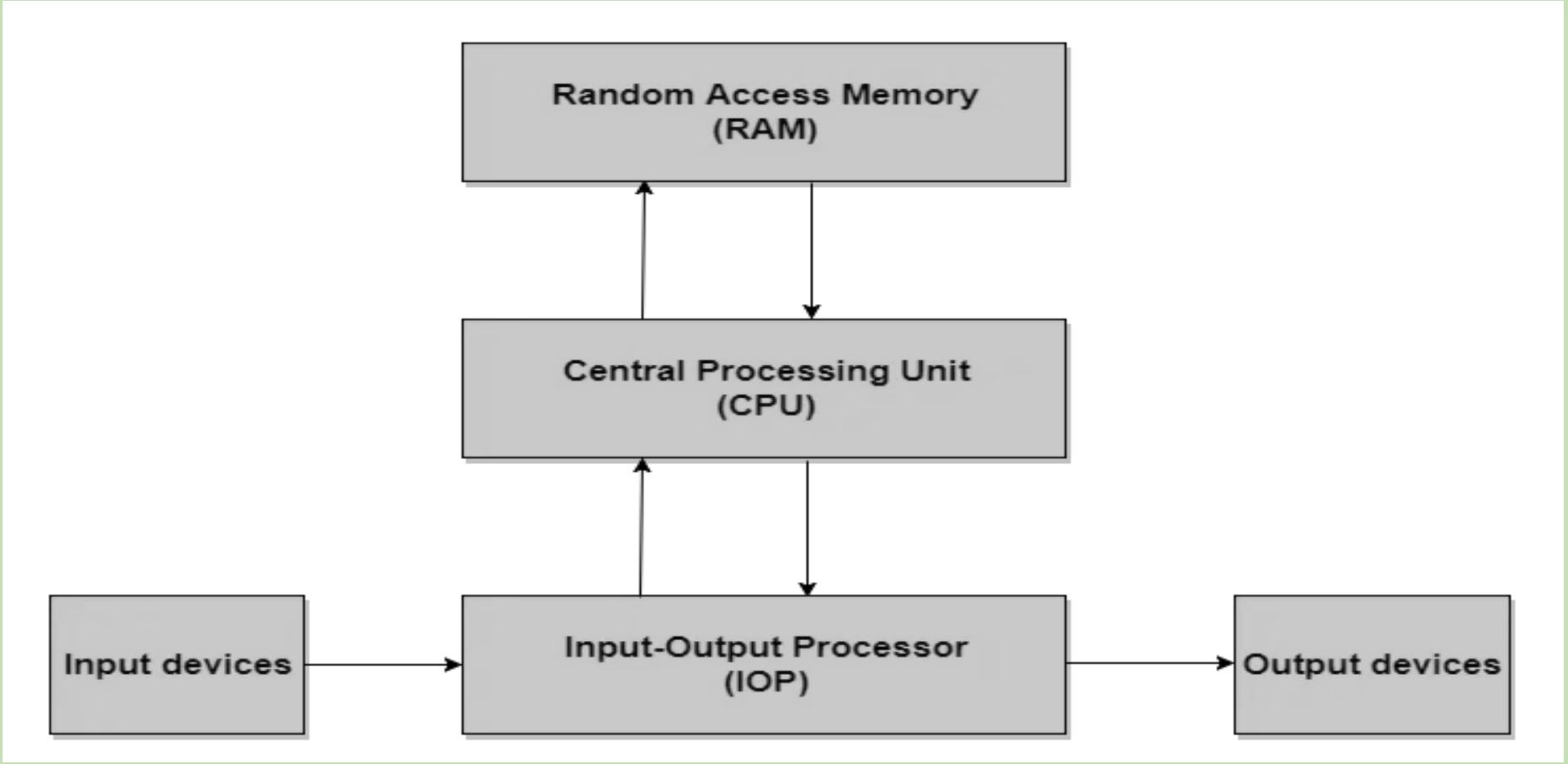
1. von Neumann architecture

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:

- **ALU:** The Arithmetic-Logic unit that performs the computer's computational and logical functions.
- **RAM:** Memory; more specifically, the computer's main, or fast, memory, also known as Random Access Memory(RAM).
- **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
- **Man-machine interfaces**; i.e. input and output devices, such as keyboard for input and display monitor for output.
- An example of computer architecture base on the von Neumann architecture is the desktop **personal computer**.

Block diagram of a Digital Computer



2. Harvard architecture

- The **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.

Basics Of Digital Components

- **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 vary from 14 to several thousand. Each pin is identified by a unique number printed on its body.

Categories of Integrated Circuits

- **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.

- **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.

- **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.

- **VLSI(Very Large Scale Integration Device)**

It contains thousands of gates within a single package such as microcomputer chip.

- **ULSI(Ultra Large Scale Integration Device)**

It contains hundred of thousands of gates within a single package such as microcomputer chip

Registers in Computer Architecture

- 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.

- A register consists of a group of flip-flops and gates. The flip-flops hold the binary information and gates control when and how new information is transferred into a register. Various types of registers are available commercially. 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:
- **Accumulator**: This is the most common register, used to store data taken out from the memory.
- **General Purpose Registers**: This is used to store data intermediate results during program execution. It can be accessed via assembly programming.
- **Special Purpose Registers**: Users do not access these registers. These registers are for Computer system,
  - **MAR:** Memory Address Register are those registers 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.

$R2 \leftarrow R1$

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 ($R2 \leftarrow R1$)

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:

- Register transfer micro-operations transfer binary information from one register to another.
- Arithmetic micro-operations perform arithmetic operations on numeric data stored in registers.
- Logic micro-operations perform bit manipulation operation on non-numeric data stored in registers.
- 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 $R3 \rightarrow R1 + R2' + 1$

- **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
$R3 \leftarrow R1 + R2$	Contents of R1+R2 transferred to R3.
$R3 \leftarrow R1 - R2$	Contents of R1-R2 transferred to R3.
$R2 \leftarrow (R2)'$	Compliment the contents of R2.
$R2 \leftarrow (R2)' + 1$	2's compliment the contents of R2.
$R3 \leftarrow R1 + (R2)' + 1$	R1 + the 2's compliment of R2 (subtraction).
$R1 \leftarrow R1 + 1$	Increment the contents of R1 by 1.
$R1 \leftarrow R1 - 1$	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.

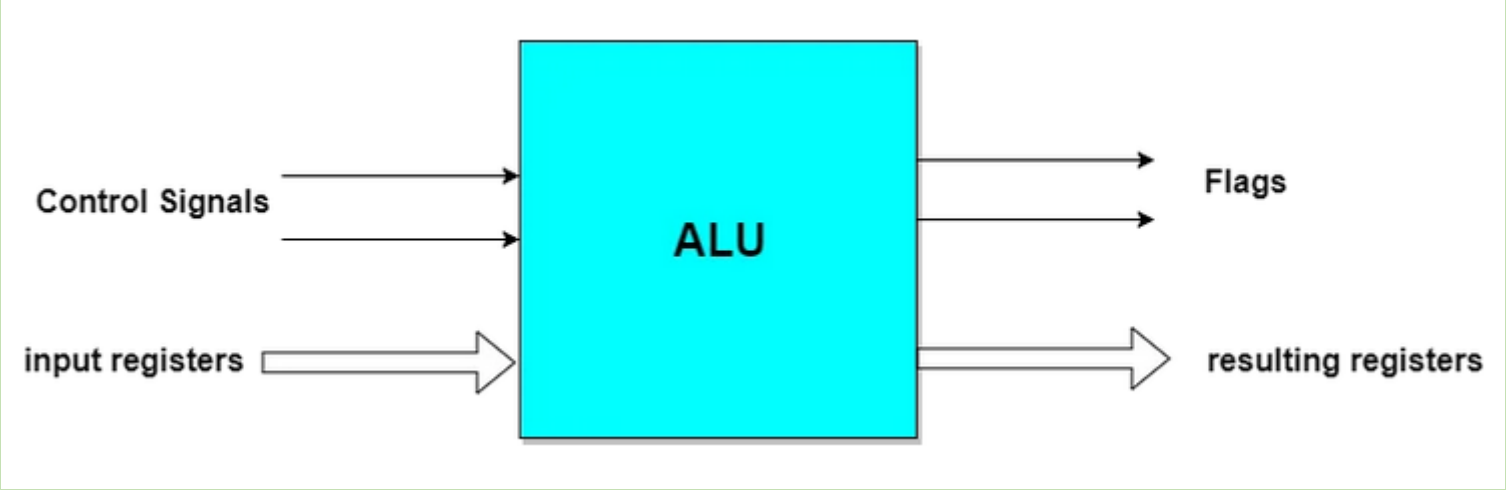
- **b) Circular Shift**

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.



Instruction Codes

- **Program,** is A set of instructions that specify the operations, operands, and the sequence by which processing has to occur. An **instruction code** is a group of bits that tells the computer to perform a specific operation part.

Instruction Code: Operation Code

The operation code of an instruction is a group of bits that define operations such as add, subtract, multiply, shift and compliment. The number of bits required for the operation code depends upon the total number of operations available on the computer. The operation code must consist of at least n bits for a given 2^n operations. The operation part of an instruction code specifies the operation to be performed.

Instruction Code: Register Part

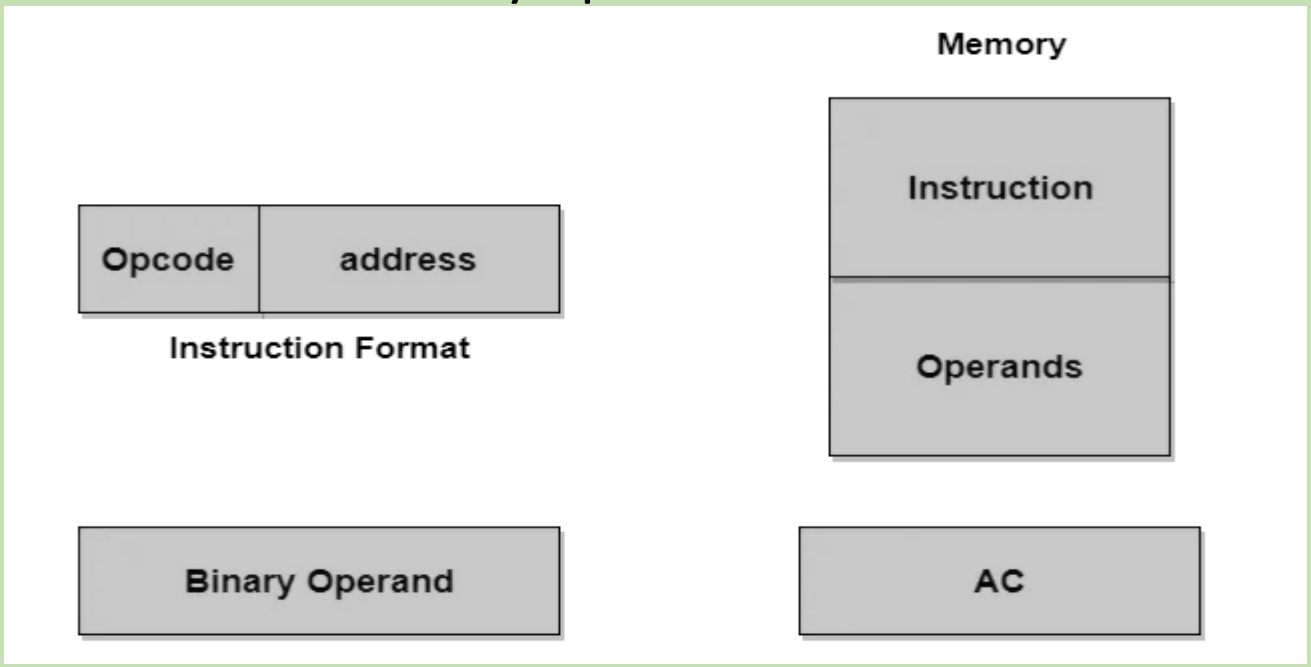
The operation must be performed on the data stored in registers. An instruction code therefore specifies not only operations to be performed but also the registers where the operands(data) will be found as well as the registers where the result has to be stored.

Stored Program Organization

Simplest way to organize a computer is to have Processor Register and instruction code with two parts. The first part specifies the operation to be performed and second specifies an address. The memory address tells where the operand in memory will be found.

Instructions are stored in one section of memory and data in another.

Computers with a single processor register is known as **Accumulator (AC)**. The operation is performed with the memory operand and the content of AC.



Common Bus System

The basic computer has 8 registers, a memory unit and a control unit. Paths must be provided to transfer data from one register to another. An efficient method for transferring data in a system is to use a Common Bus System. The output of registers and memory are connected to the common bus.

Load(LD)

The lines from the common bus are connected to the inputs of each register and data inputs of memory. The particular register whose LD input is enabled receives the data from the bus during the next clock pulse transition.

When the 2nd part of an instruction code specifies the operand, the instruction is said to have immediate operand. And when the 2nd part of the instruction code specifies the address of an operand, the instruction is said to have a direct address. And in indirect address, the 2nd part of instruction code, specifies the address of a memory word in which the address of the operand is found.

Computer Instructions

The basic computer has three instruction code formats. The Operation code (opcode) part of the instruction contains 3 bits and remaining 13 bits depends upon the operation code encountered.

There are three types of formats:

1. Memory Reference Instruction

It uses 12 bits to specify the address and 1 bit to specify the addressing mode (I). I is equal to 0 for direct address and 1 for indirect address.

2. Register Reference Instruction

These instructions are recognized by the opcode 111 with a 0 in the left most bit of instruction. The other 12 bits specify the operation to be executed.

3. Input-Output Instruction

These instructions are recognized by the operation code 111 with a 1 in the left most bit of instruction. The remaining 12 bits are used to specify the input-output operation.

Format of Instruction

The format of an instruction is depicted in a rectangular box symbolizing the bits of an instruction. Basic fields of an instruction format are given below:

- An operation code field that specifies the operation to be performed.
- An address field that designates the memory address or register.
- A mode field that specifies the way the operand of effective address is determined.

Computers may have instructions of different lengths containing varying number of addresses. The number of address field in the instruction format depends upon the internal organization of its registers.

Addressing Modes and Instruction Cycle

- 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:

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

Types of Addressing Modes

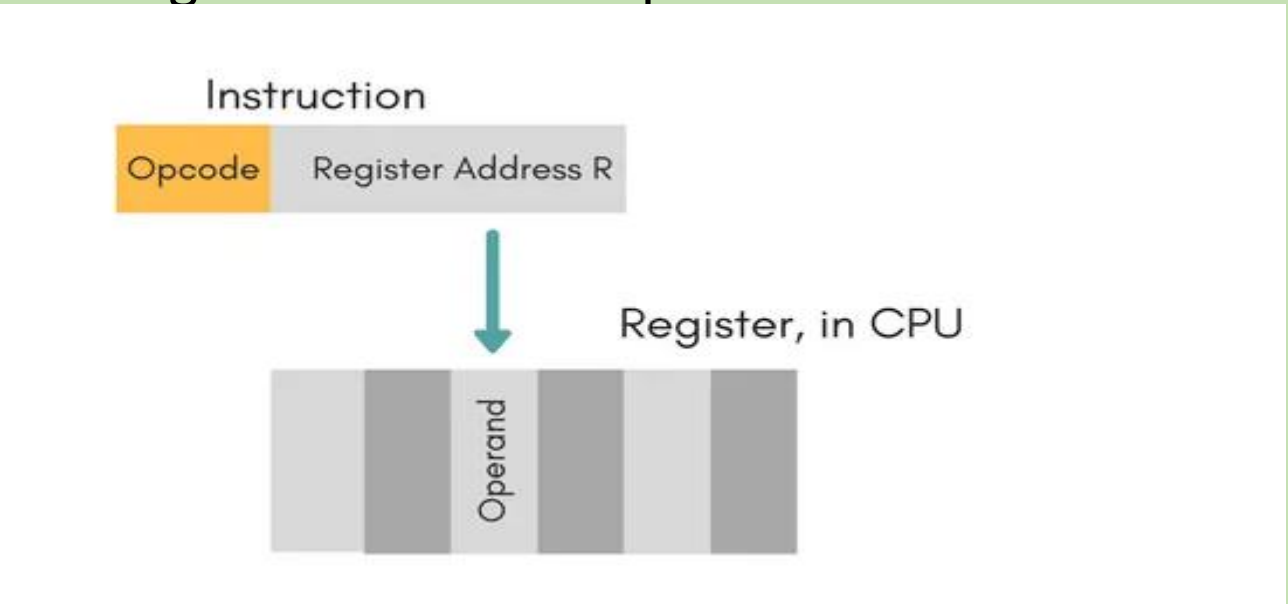
Immediate Mode

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.

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

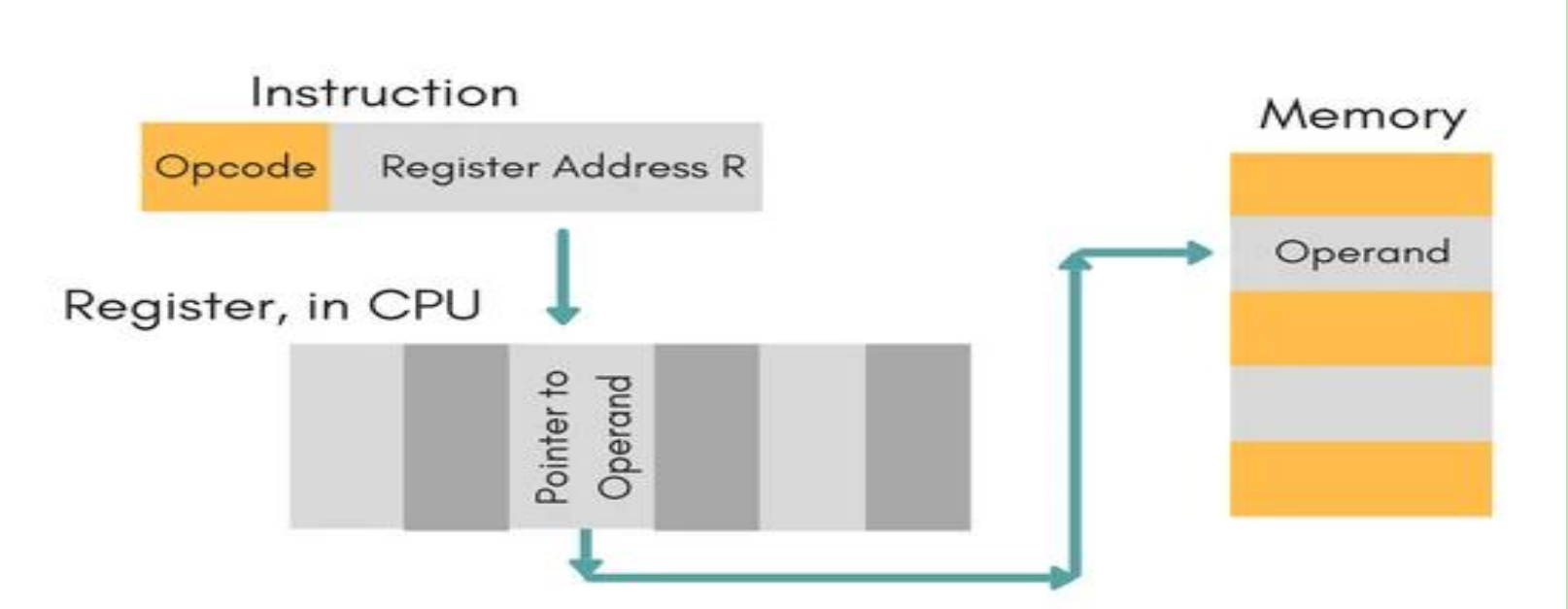
- Shorter instructions and faster instruction fetch.
- Faster memory access to the operand(s)

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

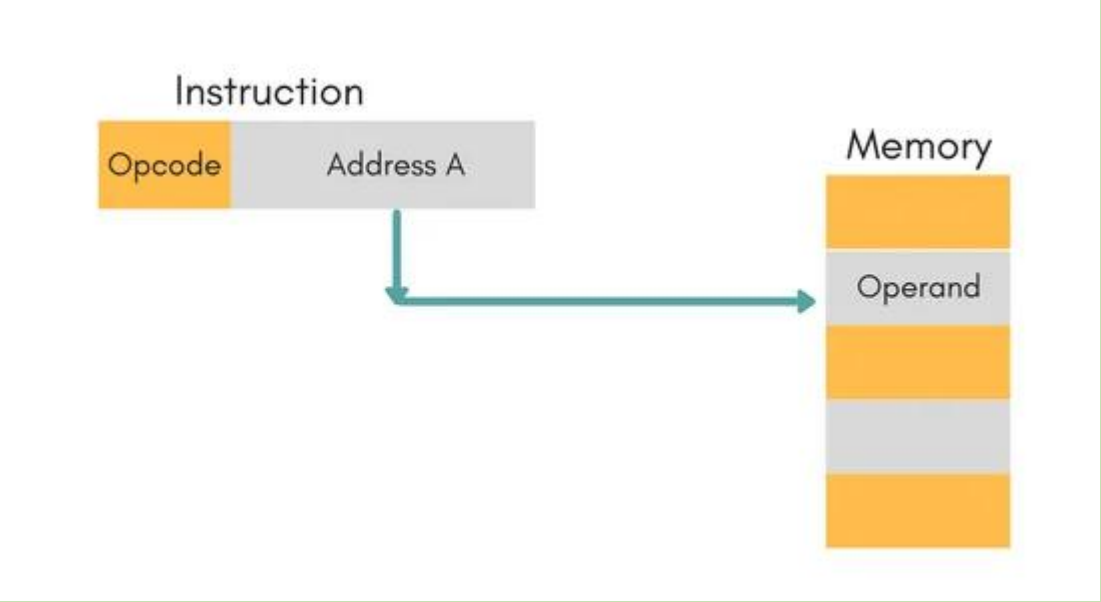
In this the register is incremented or decremented after or before its value is used.

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.

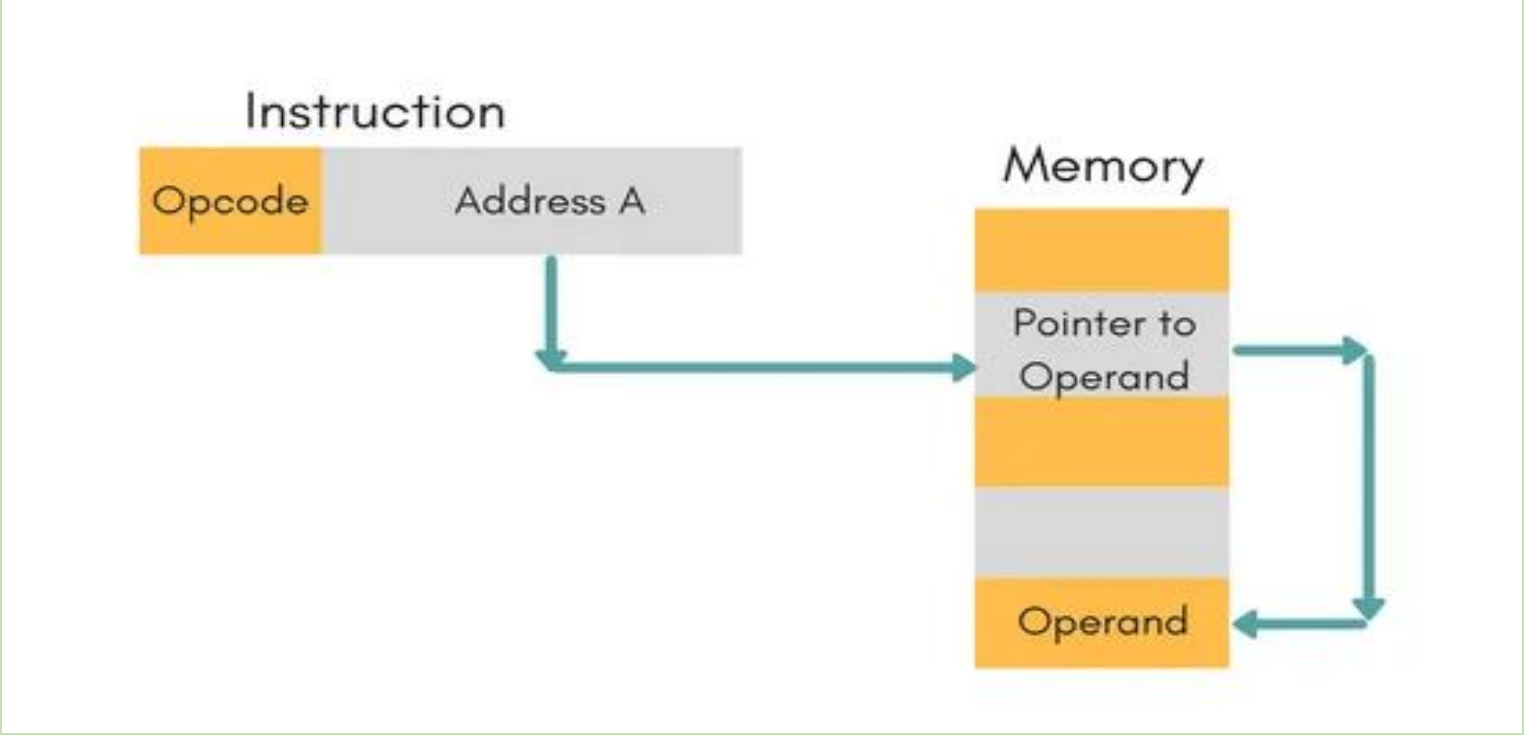
For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

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



Indirect Addressing Mode

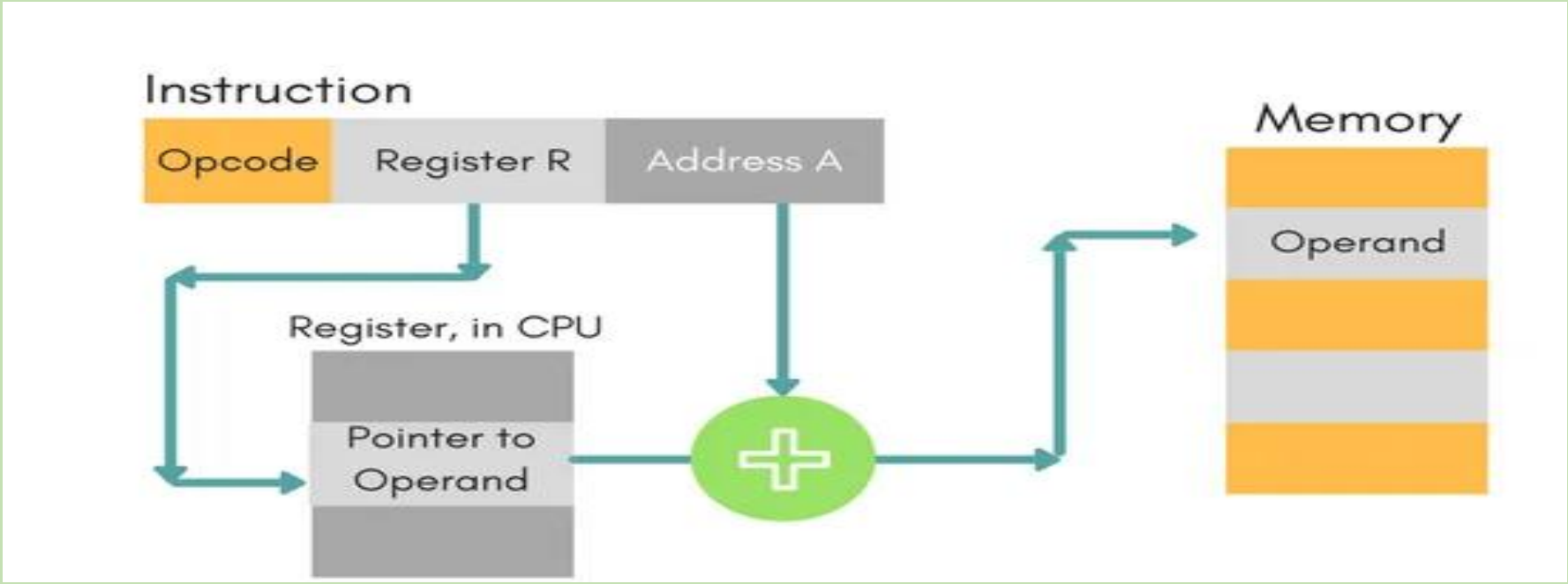
In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand.



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

It is a version of Displacement 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.

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.

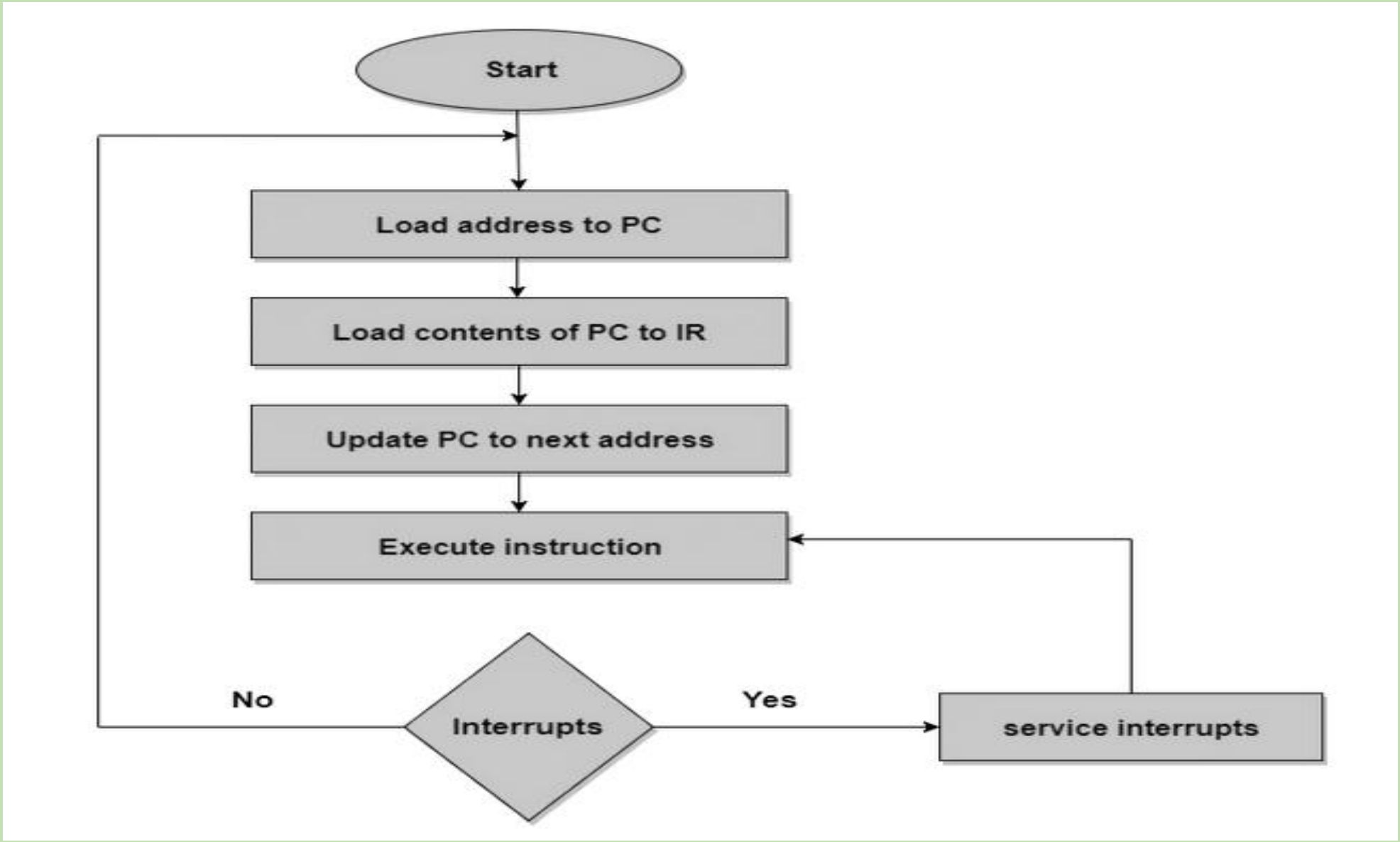
3. Read the Effective Address

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.


Start
Load address to PC
Load contents of PC to IR
Update PC to next address
Execute instruction
Interrupts
No
Yes
service interrupts