UNIT-III Central Processing Unit 8 hours

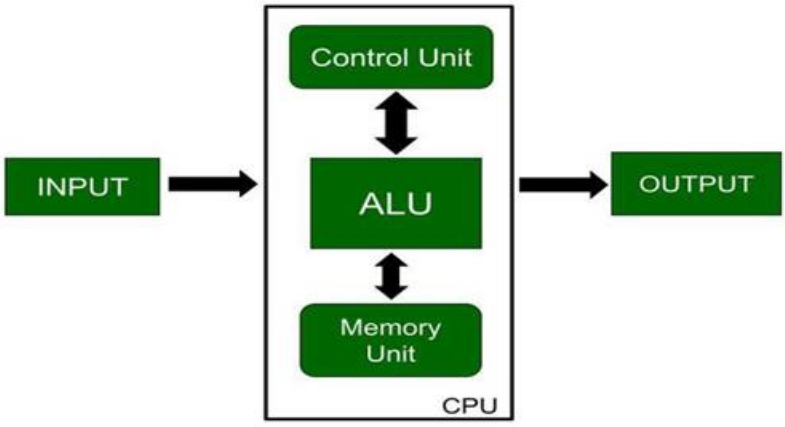
1. **Micro programmed Control Unit, Hardwired Control Unit,**
2. General register Organization, Stack Organization,
3. **Instruction types, formats, instruction cycles and sub cycles (Fetch, decode, execute etc.),**
4. execution of a complete instruction,
5. **Addressing Modes,**
6. **Reduced Instruction set computer, Complex, Instruction set Computer**

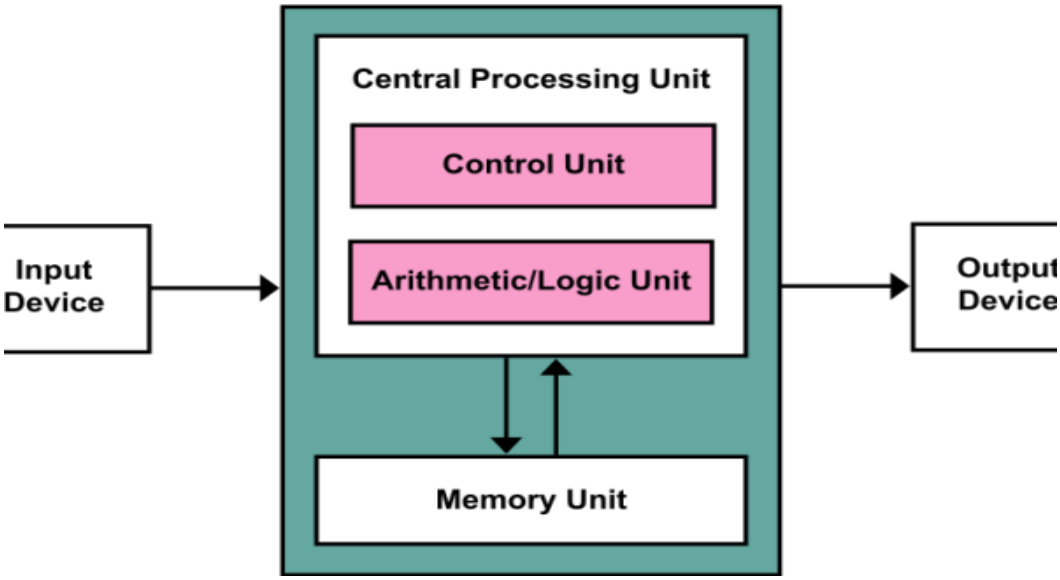
Central Processing Unit (CPU)

- CPU is the brain of the computer. All types of data processing operations and all the important functions of a computer are performed by the CPU. It helps input and output devices to communicate with each other and perform their respective operations. It also stores data which is input, intermediate results in between processing, and instructions.
- The main components of the CPU include the Control unit and the arithmetic logic unit.

Now, the CPU consists of 3 major units, which are:

1. Control Unit
2. ALU (Arithmetic Logic Unit)
3. Memory or Storage Unit



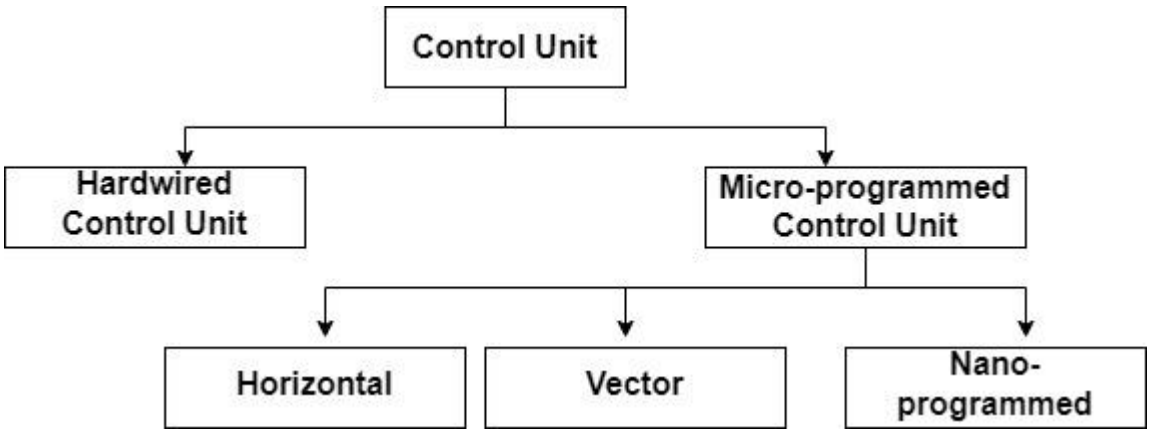


control unit performs are dependent on the type of CPU because the architecture of CPU varies from manufacturer to manufacturer. Examples of devices that require a CU are:

- Control Processing Units(CPUs)
- Graphics Processing Units(GPUs)

Control Unit

- **Control unit** works by receiving input information to which it converts into control signals, which are then sent to the central processor.
- **Control Unit** is the part of the computer's central processing unit (CPU), which directs the operation of the processor. the responsibility of the Control Unit to tell the computer's memory, arithmetic/logic unit and input and output devices how to respond to the instructions that have been sent to the processor. It fetches internal instructions of the programs from the main memory to the processor instruction register,
- **CPU** must generate the required control signal in the proper sequence. There are two approaches used for generating the control signals in proper sequence as **Hardwired Control unit** and the **Micro-programmed control unit**.



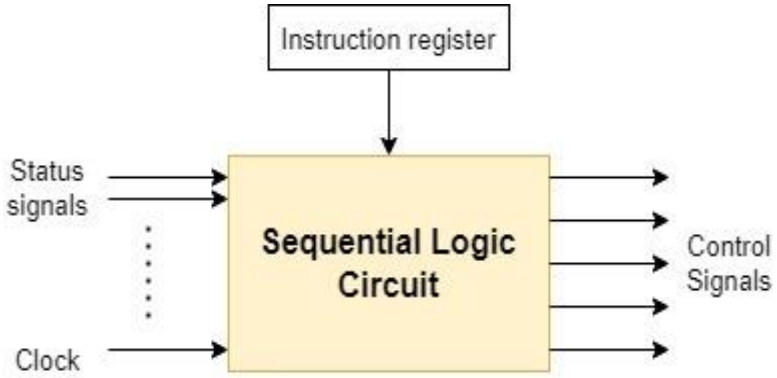
Design of Control Unit

The Control Unit is classified into two major categories:

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

Hardwired Control

- The basic design of a hardwired control unit is shown above. In this type, the control signals are generated by a special hardware **logic circuit** without any change in the structure of the circuit. In this, the generated signal cannot be modified for execution in the processor.
- The Hardwired Control organization involves the control logic to be implemented with gates, flip-flops, decoders, and other digital circuits.



Control memory is absent in this control unit. This control unit uses RISC (Reduced Instruction Set Computer) microprocessors.
Factors Considered for the design of the hardwired control unit.

- **Amount of hardware** - Minimize the number of hardware used.
- **Speed of operation** - If a single IC can replace a group of IC, replace it. The amount of hardware and speed of operation are inversely proportional to each other
- **Cost –**is too much

1. A Hard-wired Control consists of two decoders, a sequence counter, and a number of logic gates.
2. An instruction fetched from the memory unit is placed in the instruction register (IR).
3. The component of an instruction register includes; I bit, the operation code, and bits 0 through 11.
4. The operation code in bits 12 through 14 are coded with a

Designing of Hardwired Control Unit

- **State-table method** - This classical sequential design method minimises hardware and constructs a state transition table. Every generation of states has a set of control signals. This method faces the issue of desynchronization.
- **Delay Element Method** - Control signals follow a proper sequence. There is a specific time delay between two groups of control signals. D flip flops are controlled by a standard clock signal to ensure synchronisation.
- **Sequence Counter Method** - It uses counter for timing purposes.
- **PLA method** - It uses a programmable logic array to generate control signals.

Micro-programmed Control

This micro-program is a collections of micro-instructions stored in the control memory. This control unit uses CISC (Complex Instruction Set Computer) microprocessors

- A control unit whose binary control variables are stored in memory is called a Microprogrammed control unit.
- The Microprogrammed Control organization is implemented by using the programming approach.
- In Microprogrammed Control, the micro-operations are performed by executing a program consisting of micro-instructions.

Example of micro-instruction:

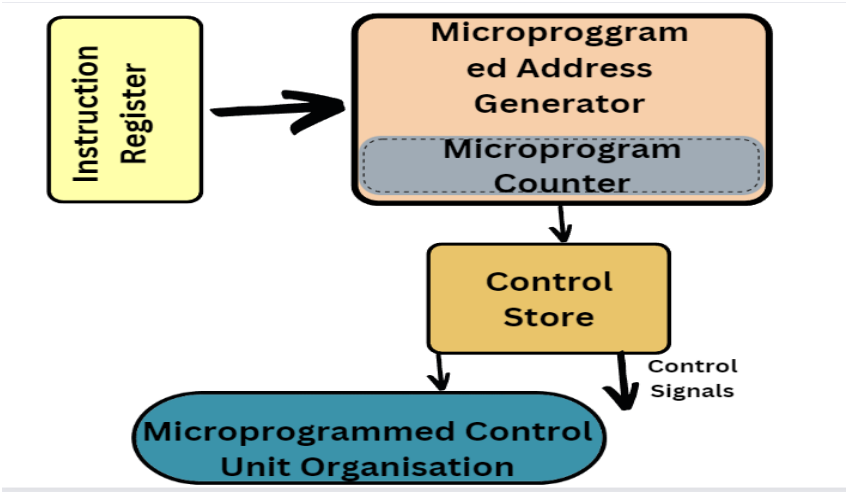
MAR←R3

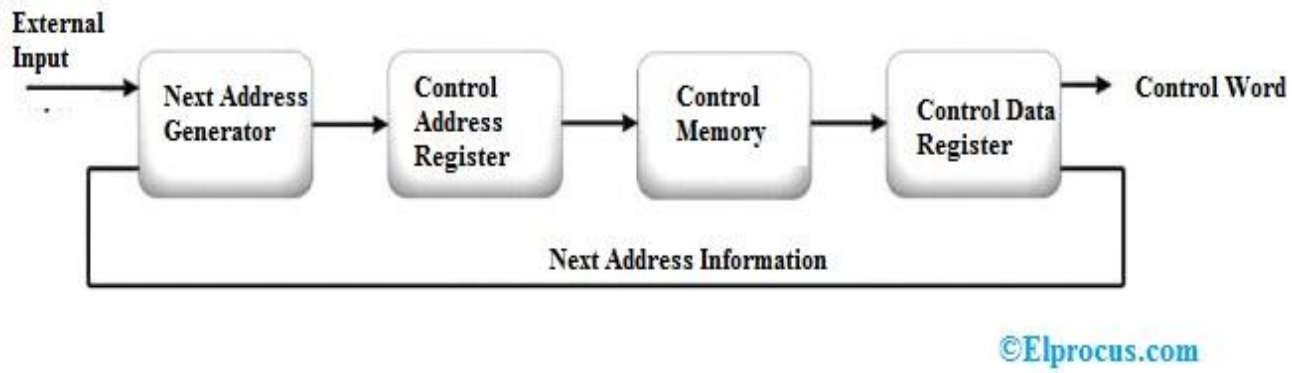
In the above instruction, we are fetching the operand.

The control signal for the above example:

MAR_{in}, $R3_{out}$

A micro-instruction consists of one or more micro-operations to be executed and the address of the next micro-instruction.





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

Characteristics of Microprogrammed Control Unit

- Micro programmes of procedures are used to implement these control units.
- The control unit used in the micro programme is a CPU nested inside of another CPU.
- These circuits are straightforward yet relatively sluggish.

Control Signals

In order to generate the control signals, both the control signals were basically designed. The functionality of a processor's hardware is operated

- Control signals are used to know what operation is going to be performed.
- It is used to know about the sequence of operations that are performed by the processor.
- It is used to know about the timing at which an operation must be executed and many other types of things.

hardwired control unit, the execution of operations is much faster, but the implementation, modification, and decoding are difficult. In contrast, implementing, modifying, decoding

micro-programmed control units are very easy. The micro-programmed control unit is also able to handle complex instructions. With the help of control signals generated by micro-programmed and hardwired control units, we are able to fetch and execute the instructions.

Hardwired Control Unit	**Microprogrammed Control Unit**
Hardwired control unit generates the control signals needed for the processor using logic circuits gates, flip-flops, decoders, and other digital circuits.	Microprogrammed control unit generates the control signals with the help of micro instructions and gates, flip-flops, decoders, and other digital circuits. stored in control memory
Hardwired control unit is faster when compared to microprogrammed control unit as the required control signals are generated with the help of hardware's	This is slower than the other as micro instructions are used for generating signals here

Hardwired Control Unit	Microprogrammed Control Unit
Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating control signals
It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions
Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated
Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)

Some Important Terms

1. **Control Word:** A control word is a word whose individual bits represent various control signals.
2. **Micro-routine:** A sequence of control words corresponding to the control sequence of a machine instruction constitutes the micro-routine for that instruction.
3. **Micro-instruction:** Individual control words in this micro-routine are referred to as microinstructions.
4. **Micro-program:** A sequence of micro-instructions is called a micro-program, which is stored in a ROM or RAM called a Control Memory (CM).

5. **Control Store:** the micro-routines for all instructions in the instruction set of a computer are stored in a special memory called the Control Store.

Computer Registers

Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as Processor registers.

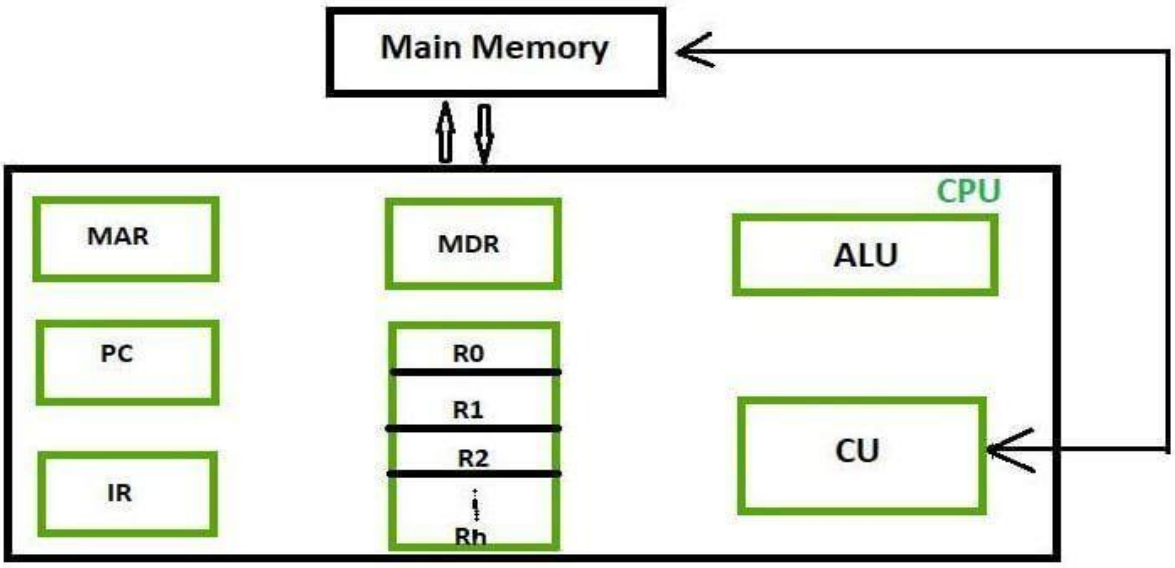
Types of Register in Computer Organization

In Computer Organisation, the register is utilized to acknowledge, store, move information and directions that are being utilized quickly by the CPU. There are different kinds of registers utilized for different reasons. Some of the commonly used registers are:

- AC (accumulator)
- DR (Data registers)
- AR (Address registers)
- PC (Program counter)
- MDR (Memory data registers)
- IR (index registers)
- MBR (Memory buffer registers)

These registers are utilized for playing out the different operations. When we perform some operations, the CPU utilizes these registers to perform the operations. When we provide input to the system for a certain operation, the provided information or the input gets stored in the registers. Once the ALU arithmetic and logical unit process the output, the processed data is again provided to us by the registers.

Operation Performed by Registers



Register	Symbol	Number of bits	Function
Data register	DR	16	Holds memory operand
Address register	AR	12	Holds address for the memory
Accumulator	AC	16	Processor register
Instruction register	IR	16	Holds instruction code
Program counter	PC	12	Holds address of the instruction
Temporary register	TR	16	Holds temporary data
Input register	INPR	8	Carries input character
Output register	OUTR	8	Carries output character

S.NO	NAME	SYMBOL	FUNCTIONING
1	Accumulator	AC	An accumulator is the most often utilized register, and it is used to store information taken from memory.
2	Memory address registers	MAR	Address location of memory is stored in this register to be accessed later. It is called by both MAR and MDR together
3	Memory data registers	MDR	All the information that is supposed to be written or the information that is supposed to be read from a certain memory address is stored here
4	General-purpose register	GPR	Consist of a series of registers generally starting from R0 and running till Rn - 1. These registers tend to store any form of temporary data that is sent to a register during any undertaking process. More GPR enables the register to register addressing, which increases processing speed.
5	Program counter	PC	These registers are utilized in keeping the record of a program that is being executed or under execution. These registers consist of the memory address of the next instruction to be fetched. PC points to the address of the next instruction to be fetched from the main memory when the previous instruction has been completed successfully. Program Counter (PC) also functions to count the number of instructions. The incrementation of PC depends on the type of architecture being used. If we use a 32-bit architecture, the PC gets incremented by 4 every time to fetch the next instruction.
6	Instructions registers	IR	Instruction registers hold the information about to be executed. The immediate instructions received from the system are fetched and stored in these registers. Once the instructions are stored in registers, the processor starts executing the set instructions, and the PC will point to the next instructions to be executed
7	Condition code registers		These have different flags that depict the status of operations. These registers set the flags accordingly if the result of operation caused zero or negative

8	Temporary registers	TR	Holds temporary data
9	Input registers	INPR	Carries input character
10	Output registers	OUTR	Carries output character
11	Index registers	BX	We use this register to store values and numbers included in the address information and transform them into effective addresses. These are also called base registers. These are used to change operand address at the time of execution, also stated as BX
12	Memory buffer register	MBR	MBR - Memory buffer registers are used to store data content or memory commands used to write on the disk. The basic functionality of these is to save called data from memory. MBR is very similar to MDR
13	Stack control registers	SCR	Stack is a set of location memory where data is stored and retrieved in a certain order. Also called last in first out (LIFO), we can only retrieve a stack at the second position only after retrieving out the first one, and stack control registers are mainly used to manage the stacks in the computer. SP - BP is stack control registers. Also, we can use DI, SI, SP, and BP as 2 byte or 4-byte registers. EDI, ESI, ESP, and EBP are 4 - byte registers
14	Flag register	FR	Flag registers are used to indicate a particular condition. The size of the registered flag is 1 - 2 bytes, and each registered flag is furthermore compounded into 8 bits. Each registered flag defines a condition or a flag. The data that is stored is split into 8 separate bits. Basic flag registers - Zero flags Carry flag Parity flag Sign flag Overflow flag.
15	Segment register	SR	Hold address for memory

16	Data register	DX	Hold memory operand

micro instruction

A single instruction in microcode. It is the most elementary instruction in the computer, such as moving the contents of a register to the arithmetic logic unit (ALU). It takes several micro instructions to carry out one complex machine instruction (CISC). Also called a "micro-op" or "µop," micro instruction

Lets take an example:

1.Suppose ADD is an instruction which is to be performed.

So it has two micro operation involved:

DR <- M [AR] - loading memory operand to data register

AC <- AC + DR - adding contents of accumulator or process register and data register and storing it to AC.

2. Suppose Load is an instruction to be performed.

It has also 2 micro operations performed:

DR <--- M [AR]

AC <--- DR

Symbols with their Binary Code for Microinstruction Fields

	Name:	Code	Symbol
F1	000	None	NOP
	001	AC ← AC + DR	ADD
	010	AC ← 0	CLRAC
	011	AC ← AC + 1	INCAC
	100	AC ← DR	DRTAC
	101	AR ← DR(0 − 10)	DRTAR
	110	AR ← PC	PCTAR

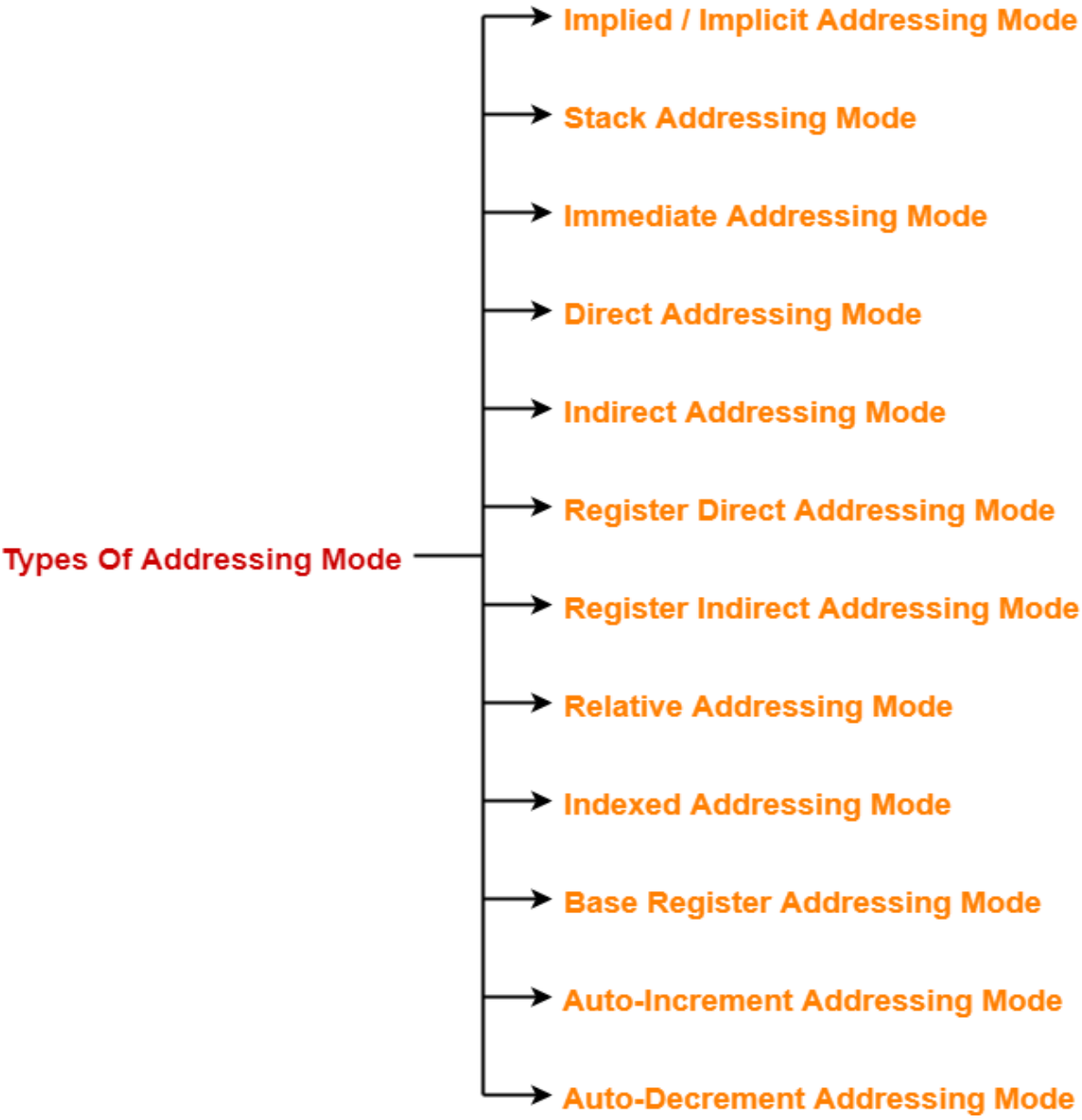
Addressing Modes

The different ways of specifying the location of an operand in an instruction are called as **addressing modes**.

The operands of the instructions can be located either in the main memory or in the CPU registers. If the operand is placed in the main memory, then the instruction provides the location address in the operand field. Many methods are followed to specify the operand address. The different methods/modes for specifying the operand address in the instructions are known as addressing modes.

Instruction format with mode field

1. To reduce the number of bits in addressing field of instruction.



1. Implied / Implicit Addressing Mode
2. Stack Addressing Mode
3. Immediate Addressing Mode
4. Direct Addressing Mode
5. Indirect Addressing Mode

6. Register Direct Addressing Mode
7. Register Indirect Addressing Mode
8. Relative Addressing Mode
9. Indexed Addressing Mode
10. Auto-Increment Addressing Mode
11. Auto-Decrement Addressing Mode

Implied / Implicit Addressing Mode

Implied Addressing Mode also known as "Implicit" or "Inherent" addressing mode

In this mode, the operands are specified implicitly in the definition of the instruction, operand (register or memory location or data) is specified in the instruction.

in this mode the operand is specified implicit in the definition of instruction.

As an example: The instruction :

"Complement Accumulator" is an Implied Mode instruction because the operand in the accumulator register is implied in the definition of instruction. In assembly language it is written as:

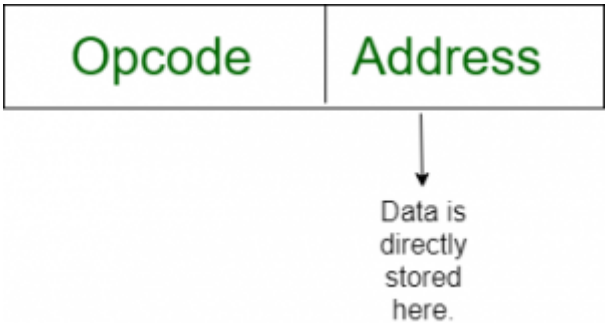
CMA: Take complement of content of AC Similarly, the instruction,

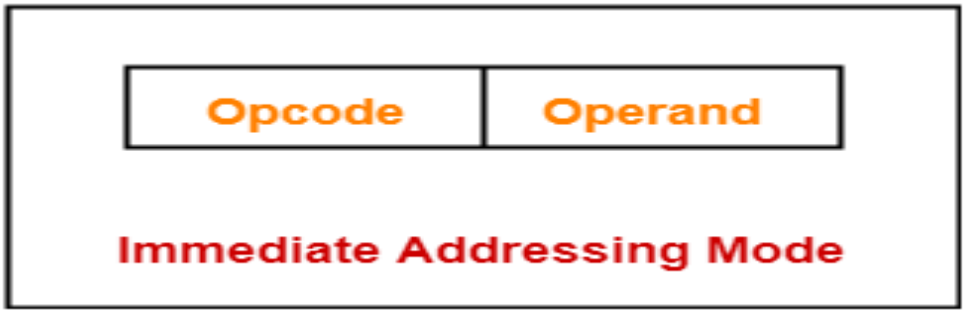
RLC: Rotate the content of Accumulator is an implied mode instruction.

Immediate Addressing Mode

Immediate addressing mode (symbol #): In this mode data is present in address field of instruction .Designed like one address instruction format.

In this mode data is present in address field of instruction .Designed like one address instruction format.
Note: Limitation in the immediate mode is that the range of constants are restricted by size of address field.






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

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

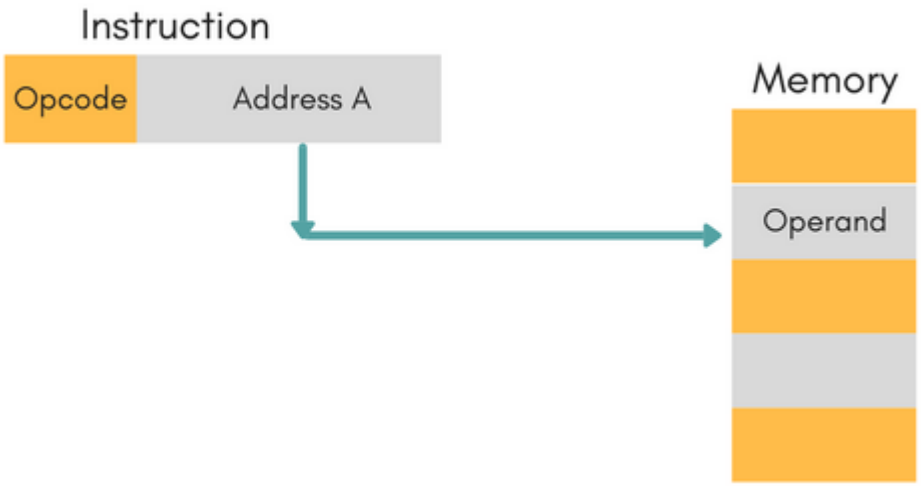
Eexample: **The Instruction:**

MVI 06	Move 06 to the accumulator
ADD 05	ADD 05 to the content of accumulator

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.

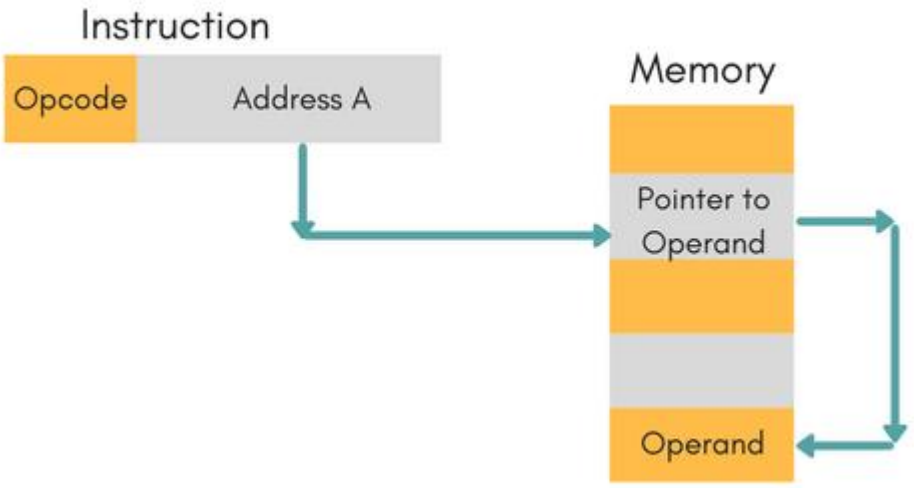


For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

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.



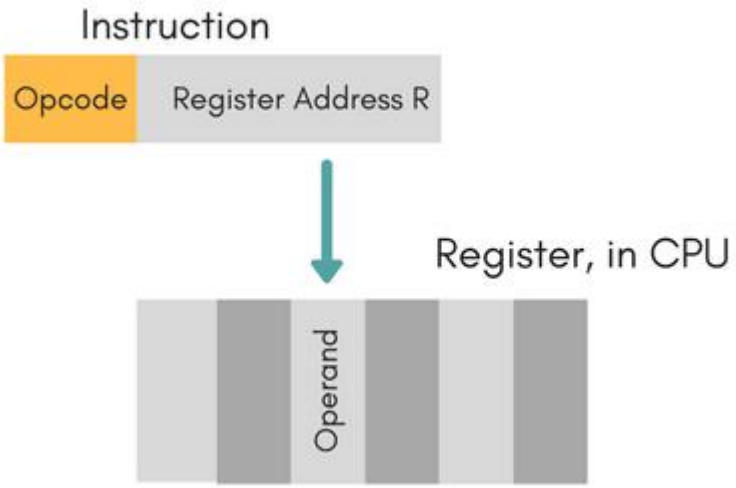
Register Addressing Mode

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.

Register Addressing Mode, the operands are in registers that reside within the CPU. That is, in this mode, instruction specifies a register in CPU, which contain the operand.It is like Direct Addressing Mode

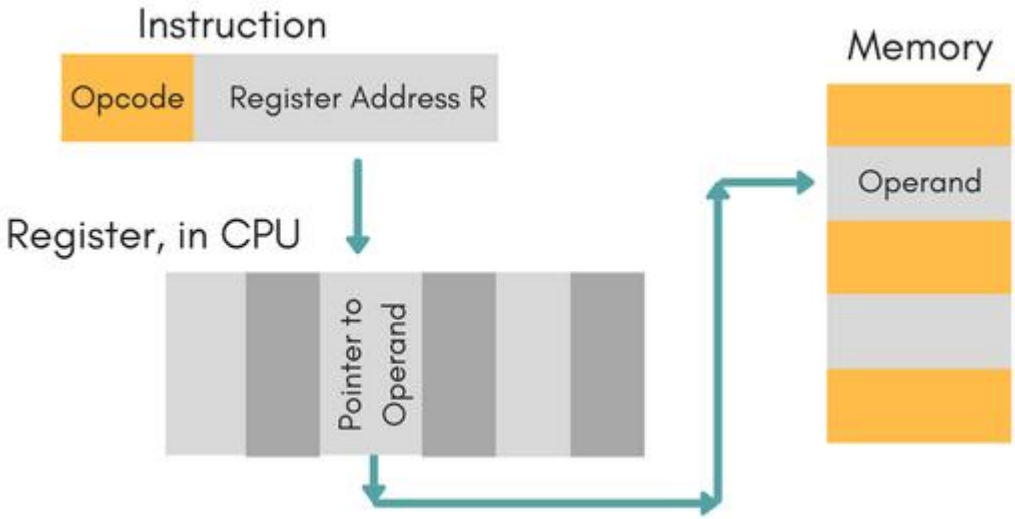
the only difference is that the address field refers to a register instead of memory location.

i.e., EA=R



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.



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.

Auto-Increment Addressing Mode-

- This addressing mode is a special case of Register Indirect Addressing Mode where-

> **Effective Address of the Operand**
>
> **= Content of Register**

In this addressing mode,

- After accessing the operand, the content of the register is automatically incremented by step size ‘d’.
- Step size ‘d’ depends on the size of operand accessed.
- Only one reference to memory is required to fetch the operand

Auto-Decrement Addressing Mode-

This addressing mode is again a special case of Register Indirect Addressing Mode where-

n this addressing mode,

- First, the content of the register is decremented by step size ‘d’.
- Step size ‘d’ depends on the size of operand accessed.
- After decrementing, the operand is read.
- Only one reference to memory is required to fetch the operand.

Effective Address of the Operand

= Content of Register – Step Size

Relative Addressing Mode:

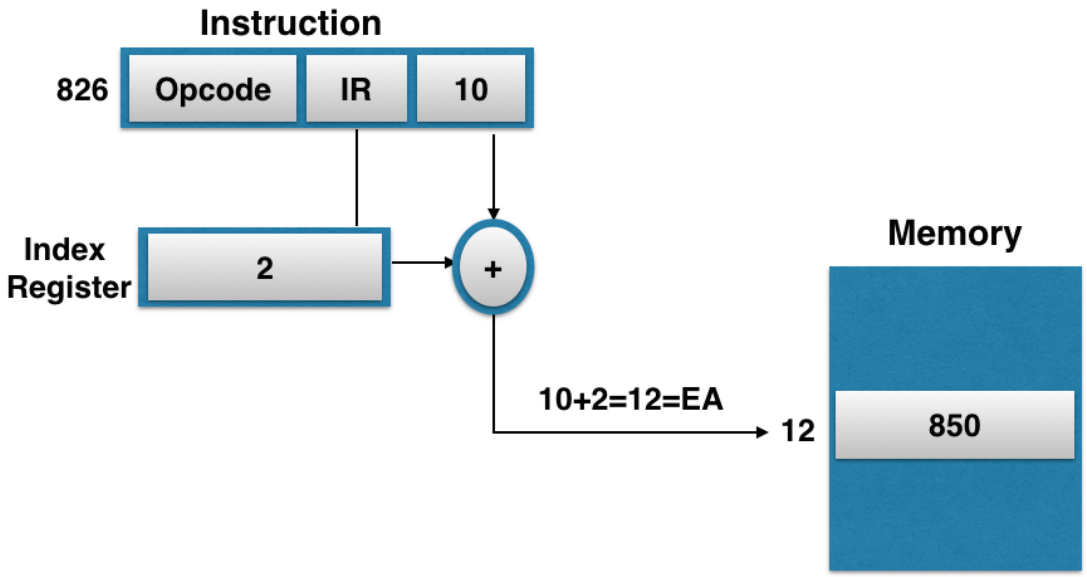
In Relative Addressing Mode , the contents of program counter is added to the address part of instruction to obtain the Effective Address.

That is, in Relative Addressing Mode, the address field of the instruction is added to implicitly reference register Program Counter to obtain effective address.

i.e., EA=A+PC

Index Register Addressing Mode:

In indexed addressing mode, the content of Index Register is added to direct address part(or field) of instruction to obtain the effective address. Means, in it, the register indirect addressing field of instruction point to Index Register, which is a special CPU register that contain an Indexed value, and direct addressing field contain base address.

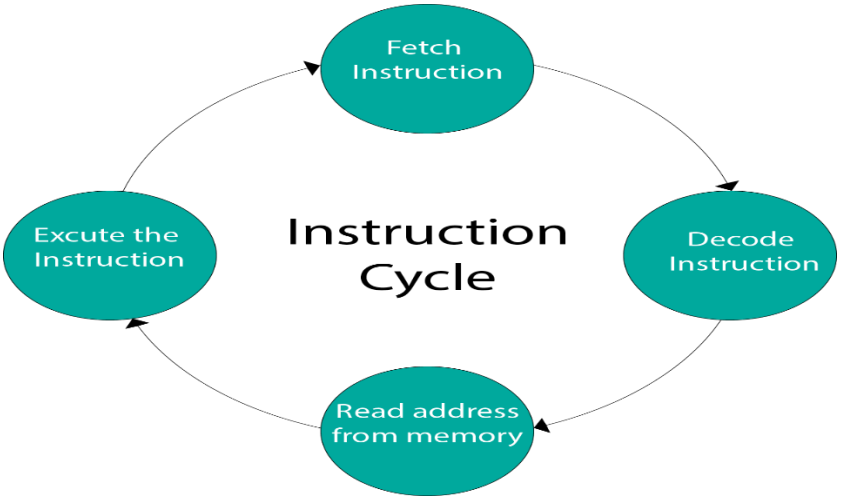


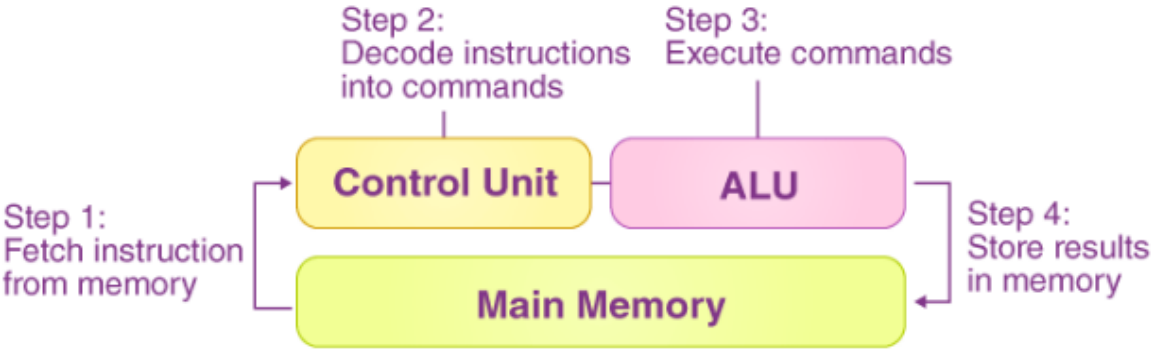
Instruction Cycle

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





new 2-bit register called *Instruction Cycle Code* (ICC). The ICC designates the state of processor in terms of which portion of the cycle it is in:-
00 : Fetch Cycle
01 : Indirect Cycle
10 : Execute Cycle
11 : Interrupt 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.

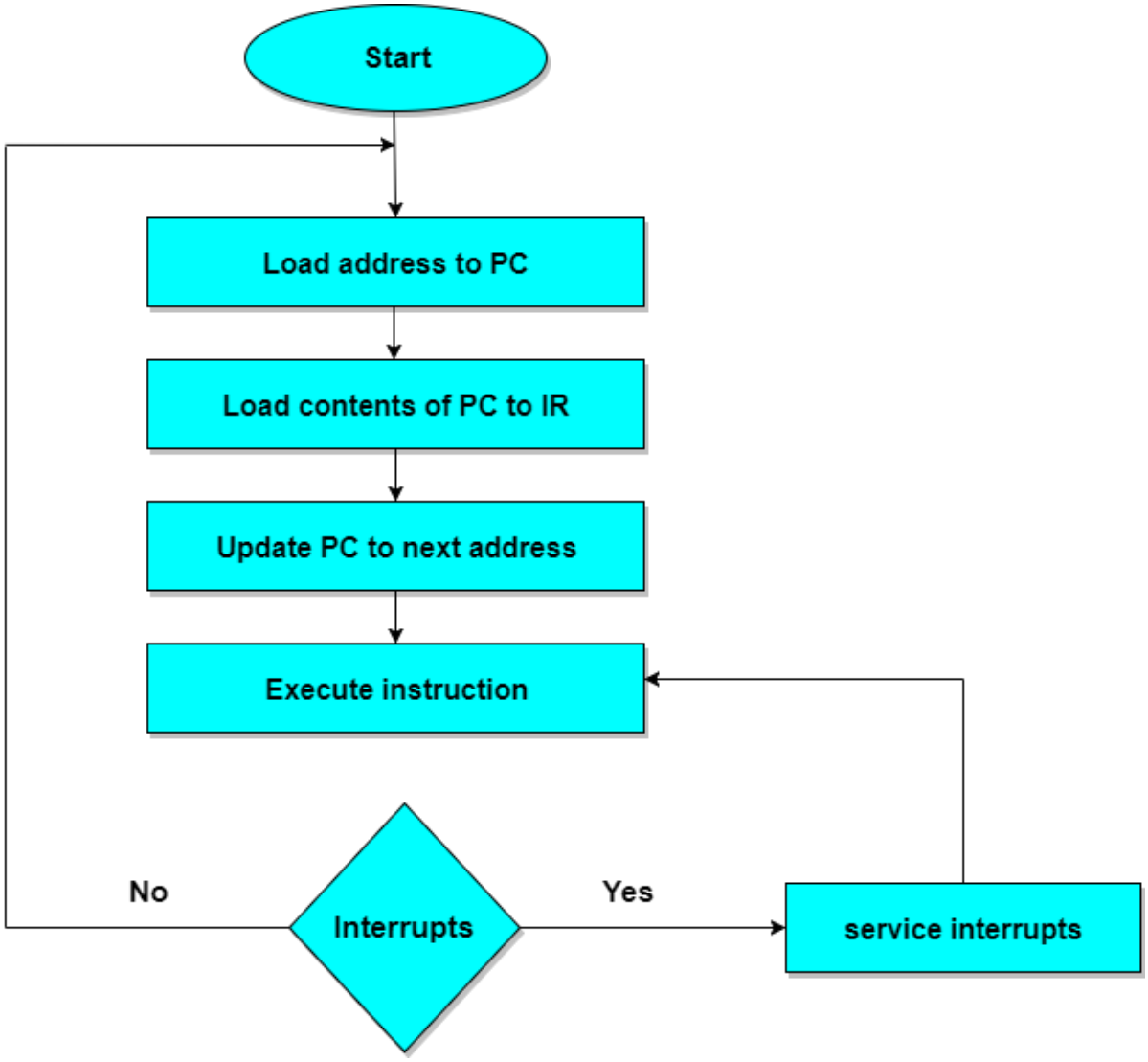
3. Read the Effective Address

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.



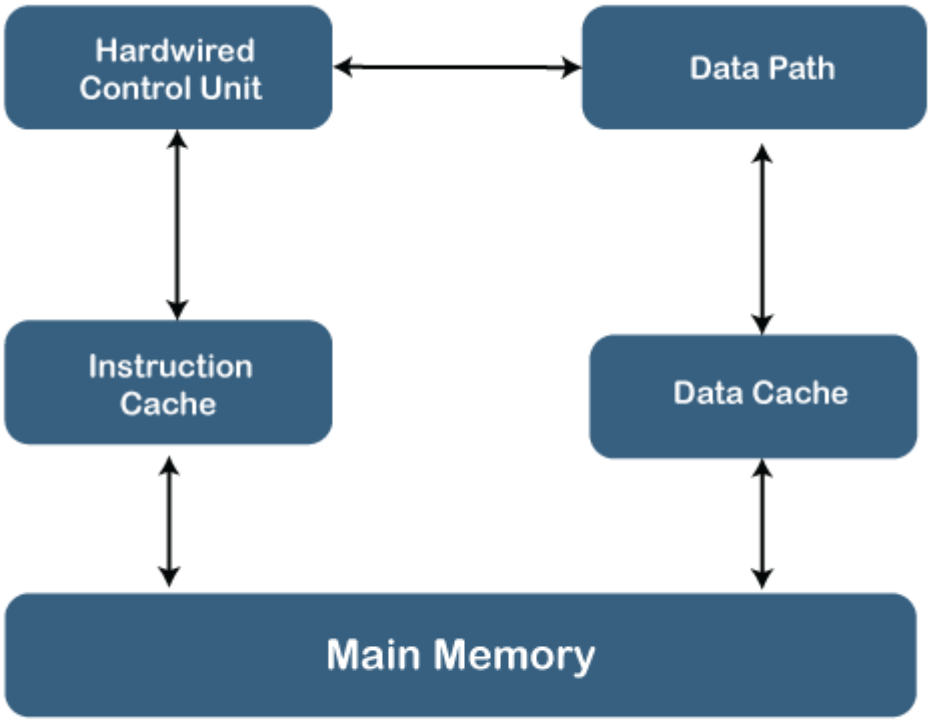
RISC Processor

RISC stands for **Reduced Instruction Set Computer Processor**, a microprocessor architecture with a simple collection and highly customized set of instructions. It is built to minimize the instruction execution time by optimizing and limiting the number of instructions. It means each instruction cycle requires only one clock cycle, and each cycle contains three parameters: fetch, decode and execute. The RISC processor is also used to perform various complex instructions by combining them into simpler ones. RISC chips require several transistors, making it cheaper to design and reduce the execution time for instruction.

Examples of RISC processors are SUN's SPARC, PowerPC, Microchip PIC processors, RISC-V.

RISC Architecture

It is a highly customized set of instructions used in portable devices due to system reliability such as Apple iPod, mobiles/smartphones, Nintendo DS,



RISC Architecture

Advantages of RISC Processor

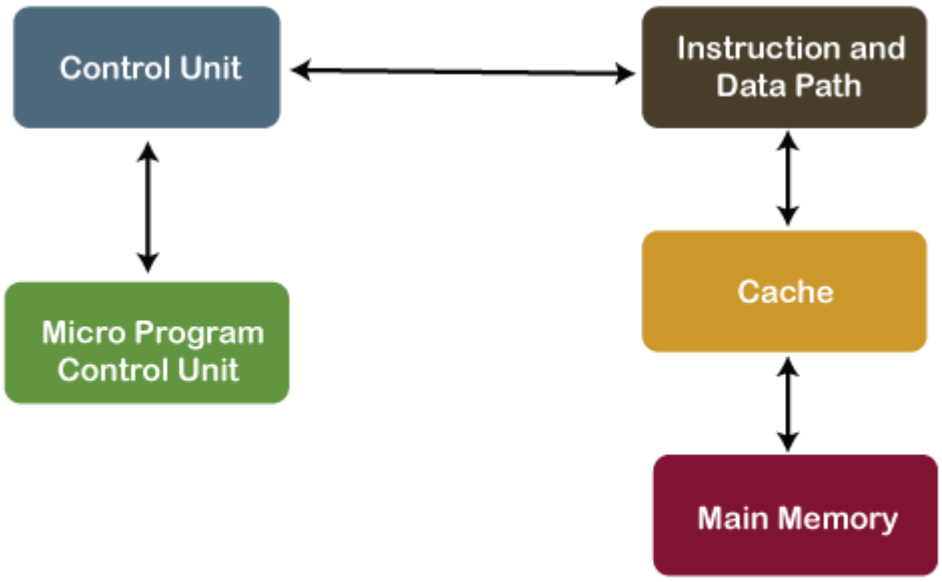
1. The RISC processor's performance is better due to the simple and limited number of the instruction set.
2. It requires several transistors that make it cheaper to design.
3. RISC allows the instruction to use free space on a microprocessor because of its simplicity.
4. RISC processor is simpler than a CISC processor because of its simple and quick design, and it can complete its work in one clock cycle.

CISC Processor

The CISC Stands for **Complex Instruction Set Computer**, developed by the Intel. It has a large collection of complex instructions that range from simple to very complex and specialized in the assembly language level, which takes a long time to execute the instructions. So, CISC approaches reducing the number of instruction on each program and ignoring the number of cycles per instruction. It emphasizes to build complex instructions directly in the hardware because the hardware is always faster than software. However, CISC chips are relatively slower as compared to RISC chips but use little instruction than RISC. Examples of CISC processors are VAX, AMD, Intel x86 and the System/360.

CISC Processors Architecture

The CISC architecture helps reduce program code by embedding multiple operations on each program instruction, which makes the CISC processor more complex. The CISC architecture-based computer is designed to decrease memory costs because large programs or instruction required large memory space to store the data, thus increasing the memory requirement, and a large collection of memory increases the memory cost, which makes them more expensive.



CISC Architecture

Characteristics of CISC Processor

Following are the main characteristics of the RISC processor:

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

Advantages of CISC Processors

1. The compiler requires little effort to translate high-level programs or statement languages into assembly or machine language in CISC processors.
2. The code length is quite short, which minimizes the memory requirement.
3. To store the instruction on each CISC, it requires very less RAM.
4. Execution of a single instruction requires several low-level tasks.
5. CISC creates a process to manage power usage that adjusts clock speed and voltage.
6. It uses fewer instructions set to perform the same instruction as the RISC.

Difference between the RISC and CISC Processors

RISC	CISC
It is a Reduced Instruction Set Computer.	It is a Complex Instruction Set Computer.
It is a hard wired unit of programming in the RISC Processor.	Microprogramming unit in CISC Processor.

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

- **RISC:** Reduce the cycles per instruction at the cost of the number of instructions per program.

- **CISC:** The CISC approach attempts to minimize the number of instructions per program but at the cost of an increase in the number of cycles per instruction.

$$CPU\ Time = \frac{Seconds}{Program} = \frac{Instructions}{Program} X \frac{Cycles}{Instructions} X \frac{Seconds}{Cycle}$$

Characteristic of RISC –

1. Simpler instruction, hence simple instruction decoding.
2. Instruction comes undersize 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.

Characteristic of CISC –

1. Complex instruction, hence 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 as operations get performed in memory itself.
5. Complex Addressing Modes.
6. More Data types.

Example – Suppose we have to add two 8-bit numbers:

- **CISC approach:** There will be a single command or instruction for this like ADD which will perform the task.
- **RISC approach:** Here programmer will write the first load command to load data in registers then it will use a suitable operator and then it will store the result in the desired location.