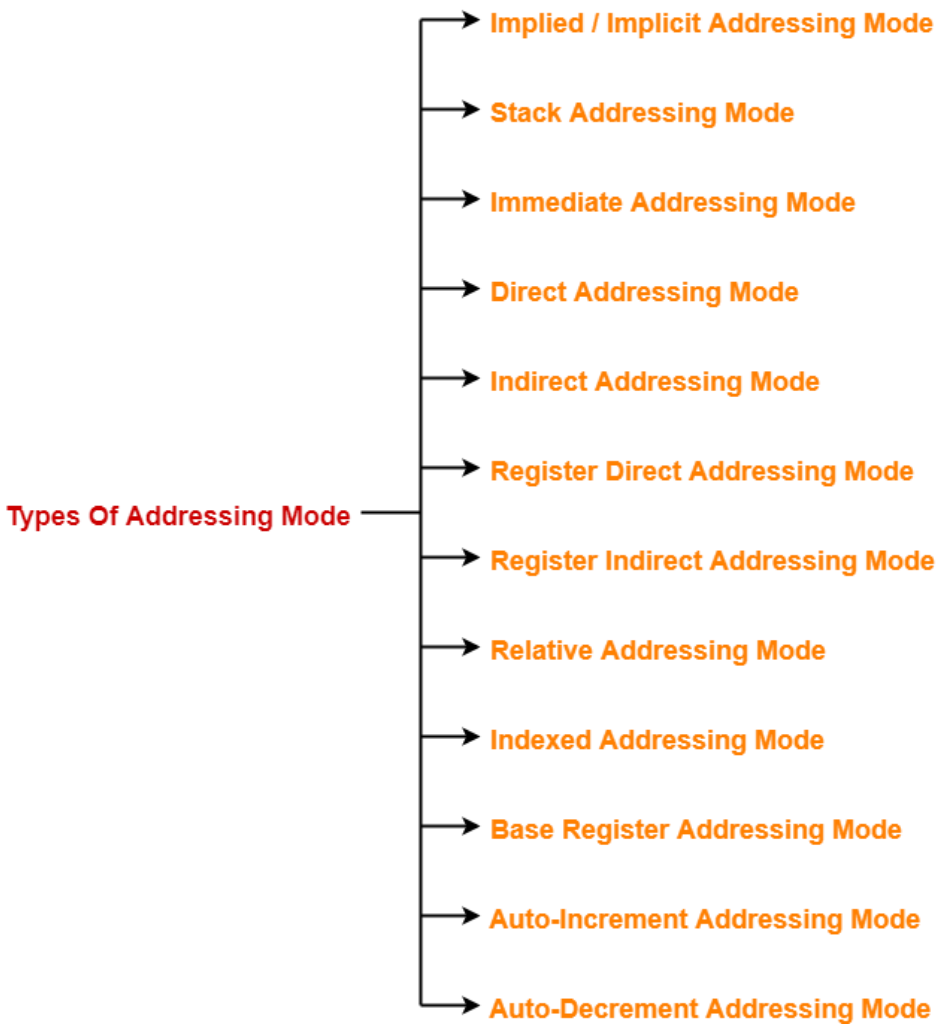
Addressing Modes-

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

Types of Addressing Modes-

In computer architecture, there are following types of addressing modes-



1. Implied Addressing Mode-

In this addressing mode,

- The definition of the instruction itself specify the operands implicitly.
- It is also called as **implicit addressing mode**.

Examples-

- The instruction “Complement Accumulator” is an implied mode instruction.
- In a stack organized computer, Zero Address Instructions are implied mode instructions.

(since operands are always implied to be present on the top of the stack)

2. Stack Addressing Mode-

In this addressing mode,

- The operand is contained at the top of the stack.

Example-

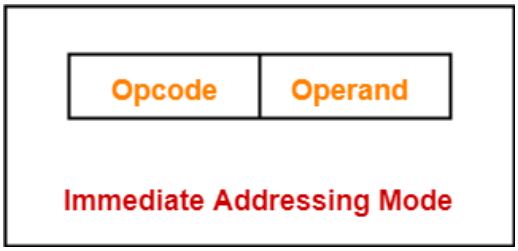
ADD

- This instruction simply pops out two symbols contained at the top of the stack.
- The addition of those two operands is performed.
- The result so obtained after addition is pushed again at the top of the stack.

3. Immediate Addressing Mode-

In this addressing mode,

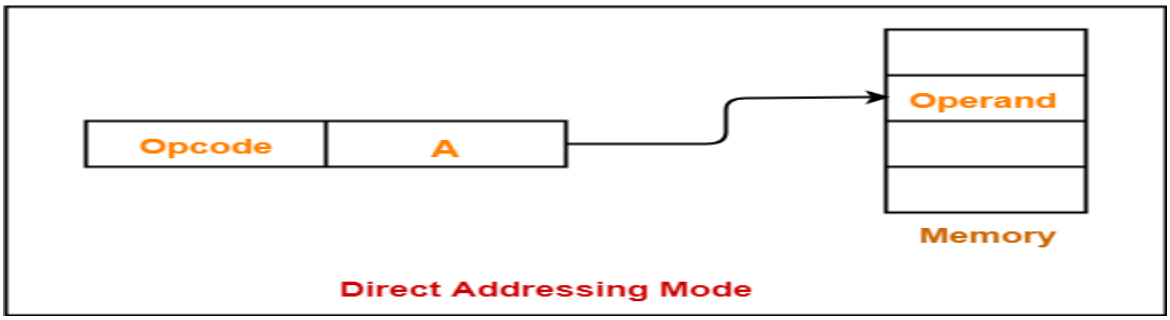
- The operand is specified in the instruction explicitly.
- Instead of address field, an operand field is present that contains the operand.
- These instructions are useful for initializing register to a constant value;
  For example MVI B, 50H



4. Direct Addressing Mode-

In this addressing mode,

- The address field of the instruction contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.
- It is also called as **absolute addressing mode**.



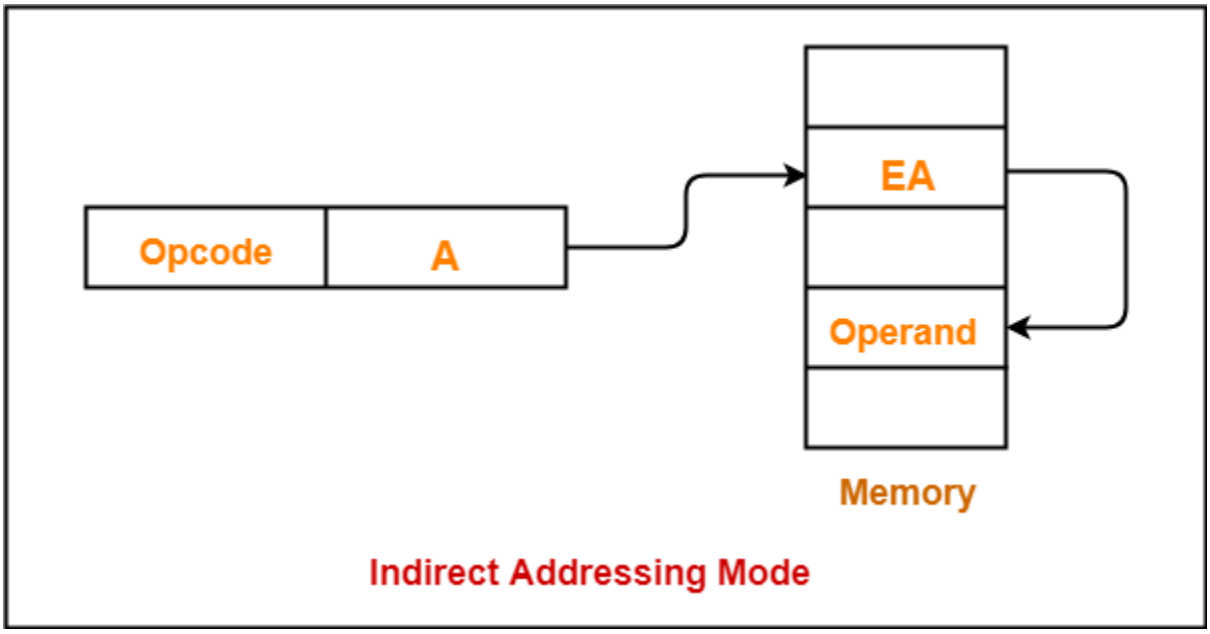
Example-

LDA 4000H

5. Indirect Addressing Mode-

In this addressing mode,

- The address field of the instruction specifies the address of memory location that contains the effective address of the operand.
- Two references to memory are required to fetch the operand.



Example-

- ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X.

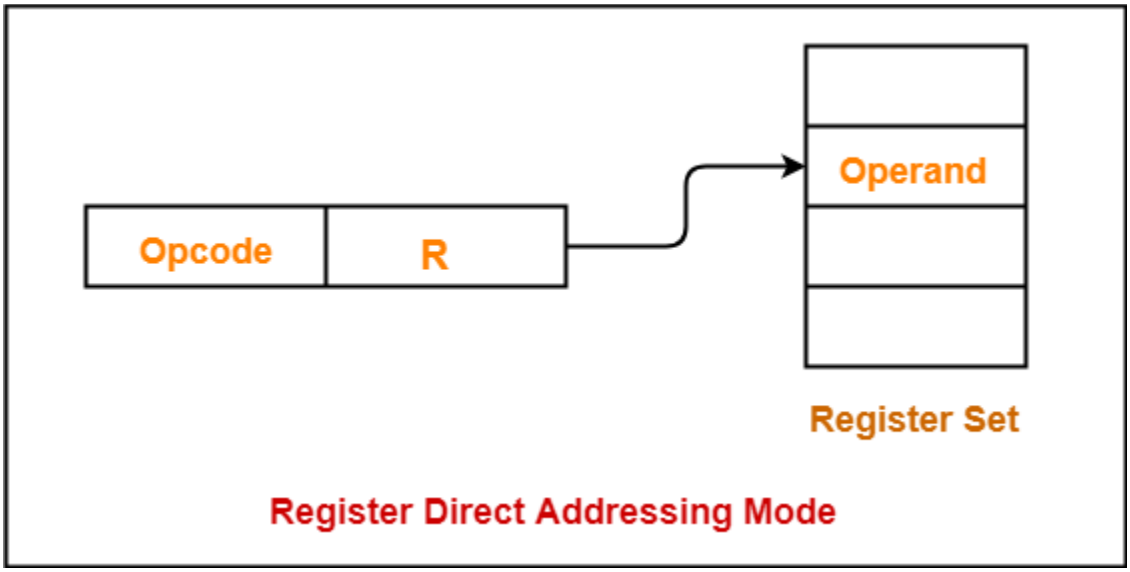
AC ← AC + [[X]]

6. Register Direct Addressing Mode-

In this addressing mode,

- The operand is contained in a register set.
- The address field of the instruction refers to a CPU register that contains the operand.
- No reference to memory is required to fetch the operand.

This addressing mode is similar to direct addressing mode. The only difference is address field of the instruction refers to a CPU register instead of main memory.



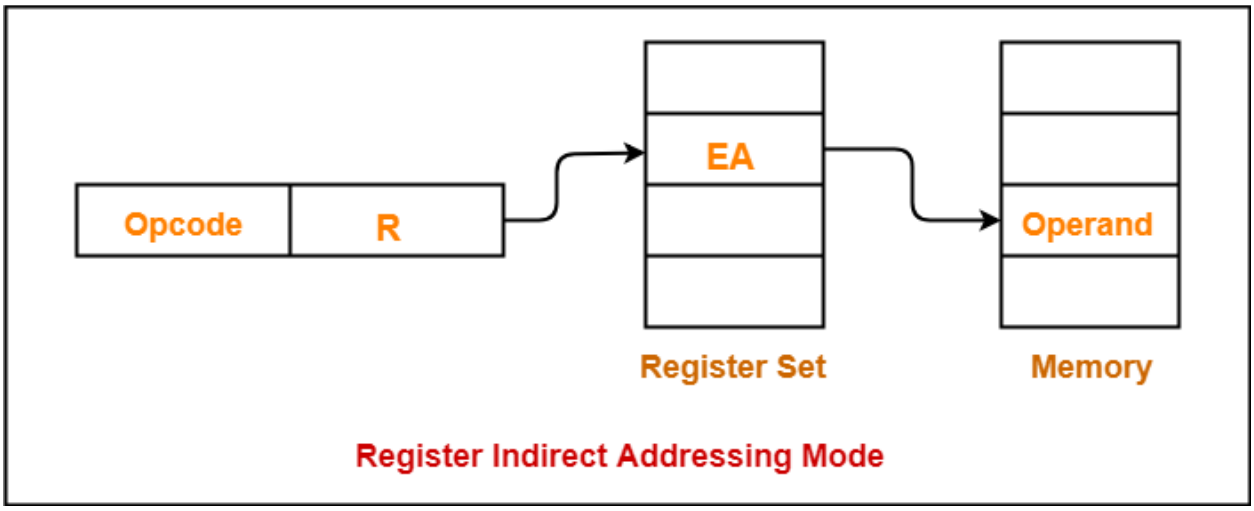
Example-

MOV A,B

7. Register Indirect Addressing Mode-

In this addressing mode,

- The address field of the instruction refers to a CPU register that contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.
- This addressing mode is similar to indirect addressing mode.
- The only difference is address field of the instruction refers to a CPU register.



Example-

LDAX B Load the accumulator indirect.

It is interesting to note-

- This addressing mode is similar to indirect addressing mode.
- The only difference is address field of the instruction refers to a CPU register.

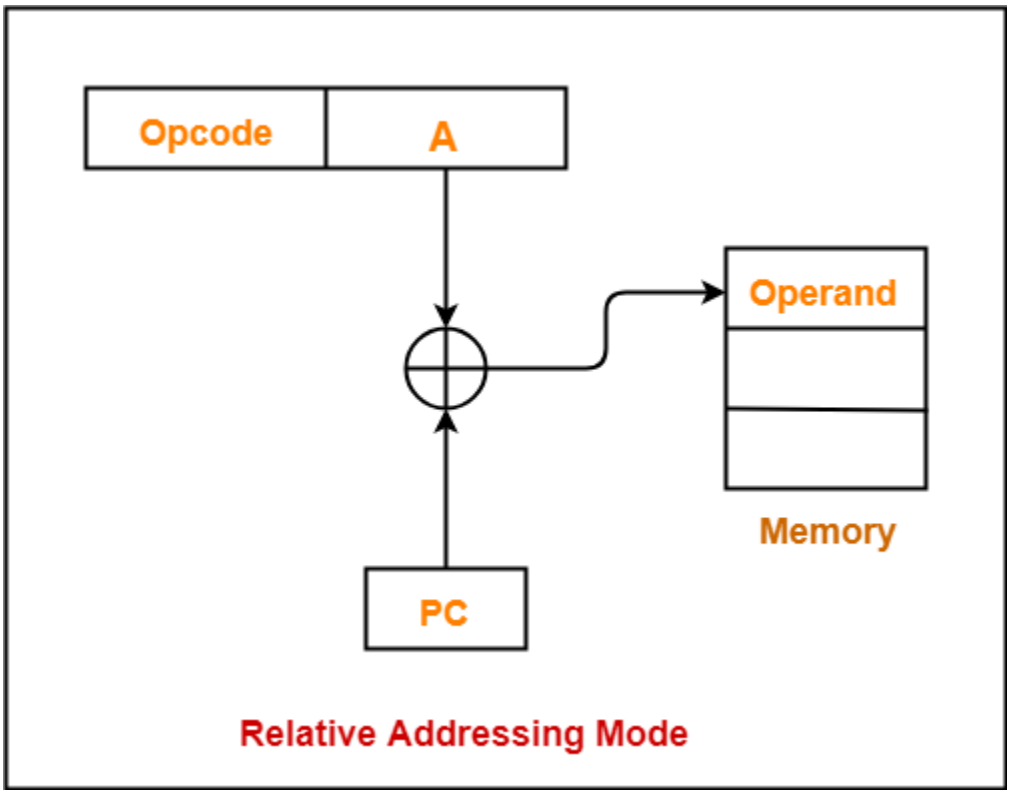
8. Relative Addressing Mode-

In this addressing mode,

- Effective address of the operand is obtained by adding the content of program counter with the address part of the instruction.
- **Program counter** (PC) always contains the address of the next instruction to be executed.
- After fetching the address of the instruction, the value of program counter immediately increases irrespective of whether the fetched instruction has completely executed or not.

Effective Address

= Content of Program Counter + Address part of the instruction



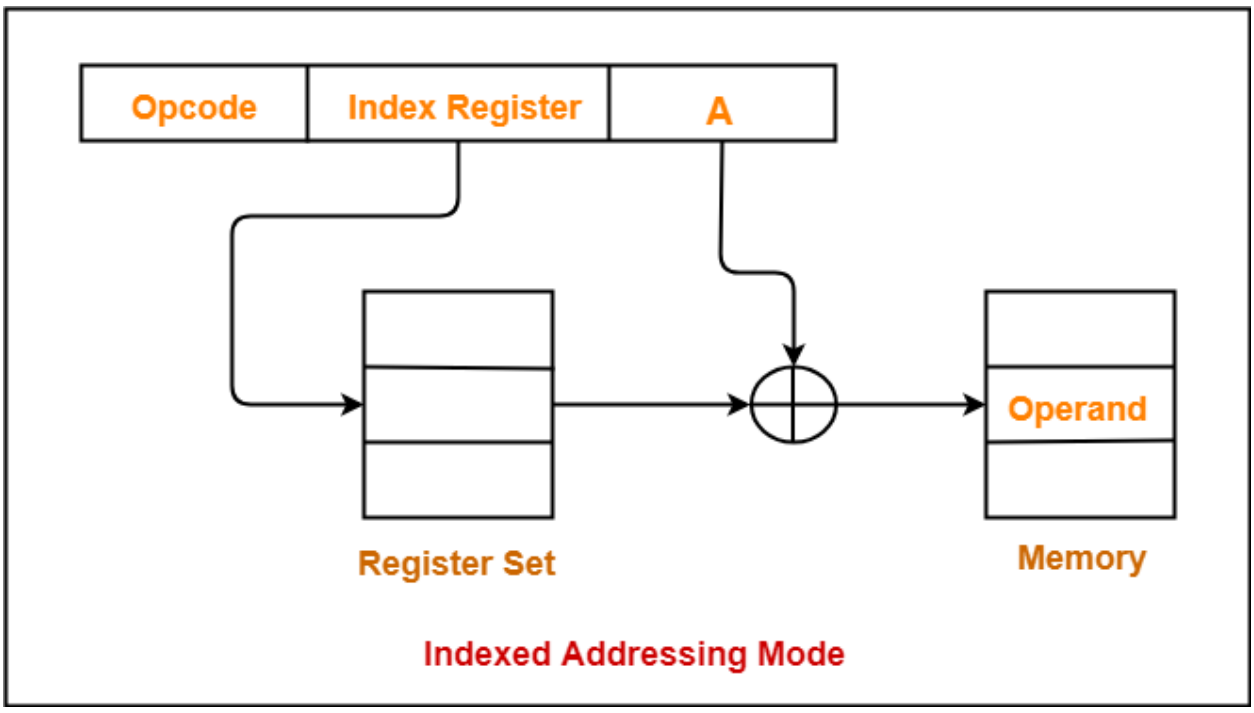
9. Indexed Addressing Mode-

In this addressing mode,

- Effective address of the operand is obtained by adding the content of index register with the address part of the instruction.

Effective Address

= Content of Index Register + Address part of the instruction



Indexed Addressing Mode

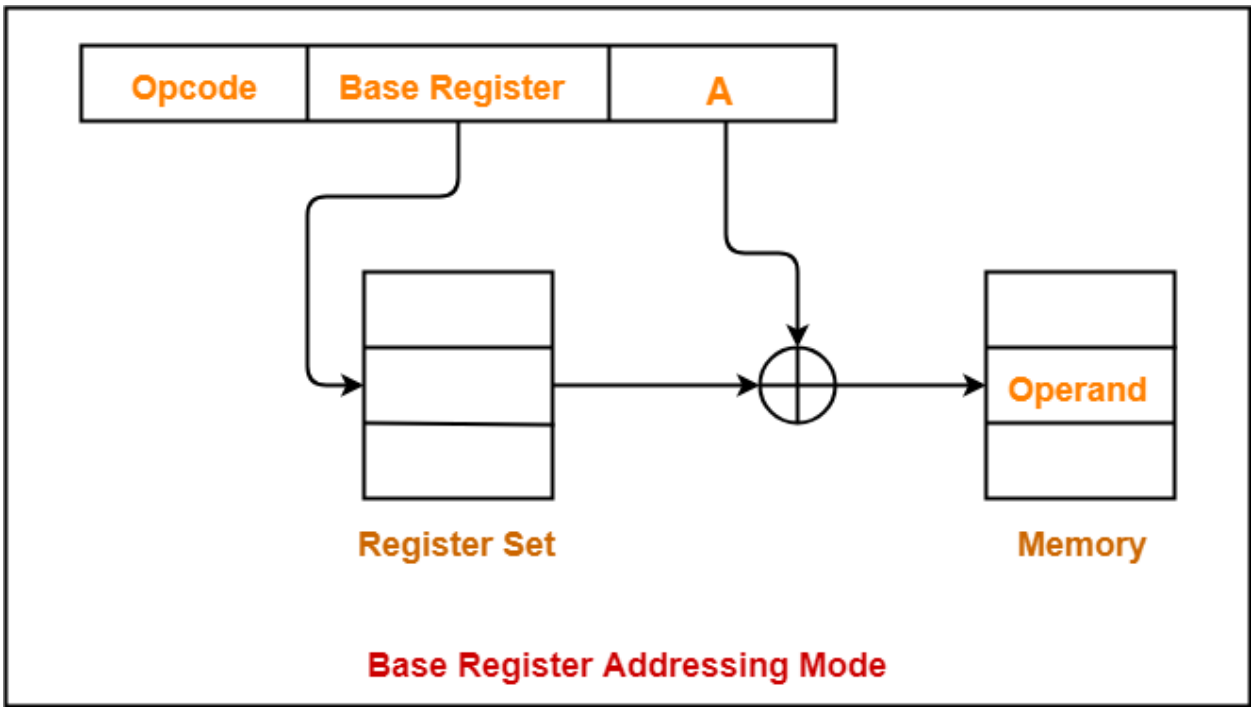
10. Base Register Addressing Mode-

In this addressing mode,

- Effective address of the operand is obtained by adding the content of base register with the address part of the instruction.

Effective Address

= Content of Base Register + Address part of the instruction



Base Register Addressing Mode

11. 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.

12. Auto-Decrement Addressing Mode-

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

Effective Address of the Operand

= Content of Register – Step Size

In 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.

Applications of Addressing Modes-

Addressing Modes	Applications
Immediate Addressing Mode	• To initialize registers to a constant value
Direct Addressing Mode and Register Direct Addressing Mode	• To access static data • To implement variables
Indirect Addressing Mode and Register Indirect Addressing Mode	• To implement pointers because pointers are memory locations that store the address of another variable • To pass array as a parameter because array name is the base address and pointer is needed to point the address
Relative Addressing Mode	• For program relocation at run time i.e. for position independent code • To change the normal sequence of execution of instructions • For branch type instructions since it directly updates the program counter
Index Addressing Mode	• For array implementation or array addressing • For records implementation

Base Register Addressing Mode	• For writing relocatable code i.e. for relocation of program in memory even at run time • For handling recursive procedures
Auto-increment Addressing Mode and **Auto-decrement Addressing Mode**	• For implementing loops • For stepping through arrays in a loop • For implementing a stack as push and pop