Addressing Modes :

Instruction format with mode field

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in computer registers or memory words.

The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Types :

1. Implied Mode.
2. Immediate mode.
3. Register Mode.
4. Register Indirect Mode.
5. Auto increment Addressing mode.
6. Auto decrement Addressing mode.
7. Direct address Mode.
8. Relative address Mode.
9. Indexed address Mode.
10. Indirect address Mode.
11. Base register address Mode.
12. Stack addressing Mode.

① Implied Mode : { Implicit Mode }

Operand is specified implicitly in the definition of instruction.

Used for zero address & one address instructions

Eg: 1. INCA ↳ { AC ← AC+1 } Increment the value of AC.

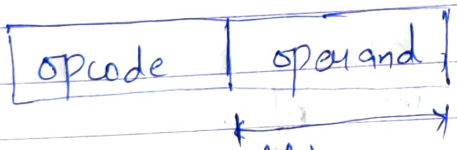
2. CRC { Clear flag }

3. CMA { Complement Accumulator }

Zero-address instructions in a stack-organized computer are implied-mode instructions since the operands are implied to be on top of the stack.

Instruction size will also decrease.

② Immediate mode : { Denoted by # }



Eg: ADD R1, #50

↳ Operand is directly provided as constant

↳ No computation required to calculate effective add

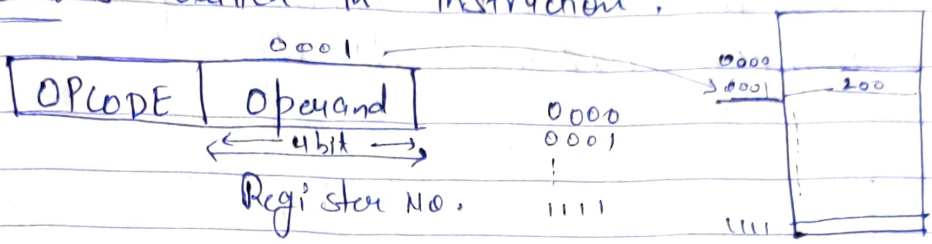
Limitation : The size of constant should be equal to the size of address field.

③ **Register Mode :**

4 bits = 16 registers
[K bit → 2^K registers]

operand is present in the register.

Register no. is written in instruction.



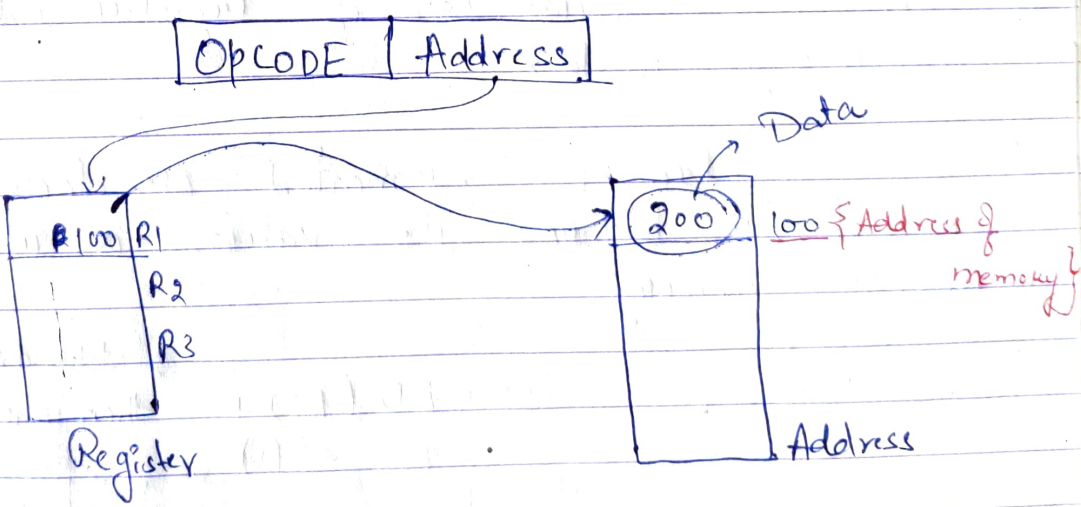
Effective Address = (R_1)

Eg: LD R1 $(AC \leftarrow R1)$

Speed will be high as we are using registers here.

④ **Register Indirect Mode :**

Register contains address of operand rather than operand itself.

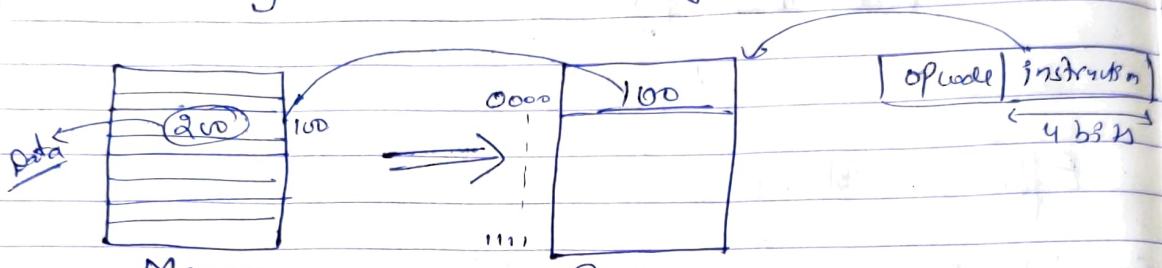


Eg: 1. LD (R1) $\{ AC \leftarrow M[(R1)] \}$

2. ADD R1, (R_2) $\{ R1 \leftarrow R1 + M[(R2)] \}$

Effective Address : The memory address obtained from the computation dictated by the given addressing mode.

Advantage : The address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.



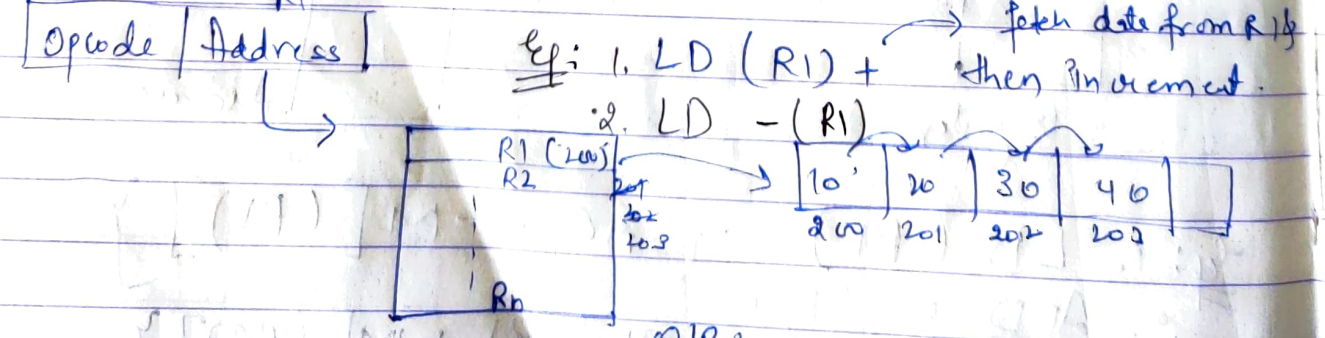
$2^6 \cdot 2^{10} = 2^{16}$ → 16 bit to (store in instruction field)

Size = 16 register → 2^4 → 4 bits in the instruction field.

(5) Auto increment or Auto decrement Addressing Mode:

This is a special case of register indirect addressing mode.

The register is incremented or decremented after (or before) its value is used to access memory.



Eg: 1. LD (R1)+ → fetch data from R1 then increment.

2. LD -(R1)

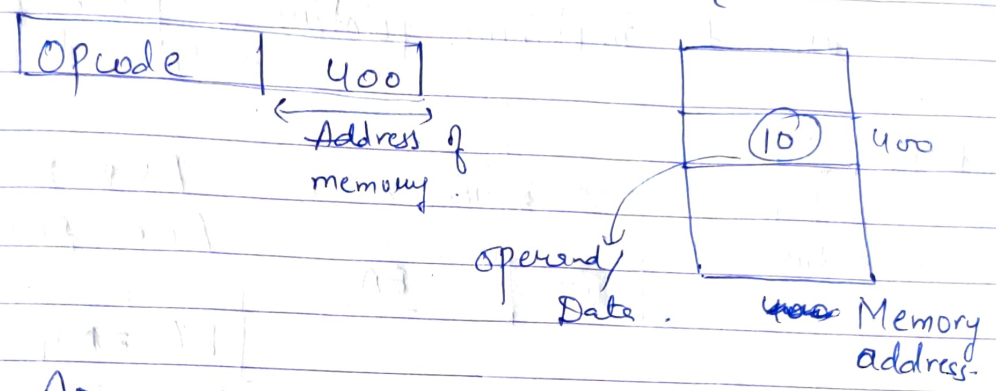
O/P: 10, 20, 30, 40.

6. **Direct Address Mode :** Symbol → [ ]

(Absolute addressing mode)

The effective address is equal to the address present in the instruction.
i.e. Actual address is given in instruction.

Use to access variables.

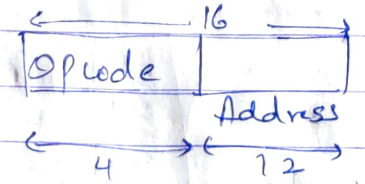


Eg: ADD X
AC ← AC + M[X]

LD [500]
AC ← M[500]

Disadvantage :

Limited value of address field.



(0 - 4095) only.

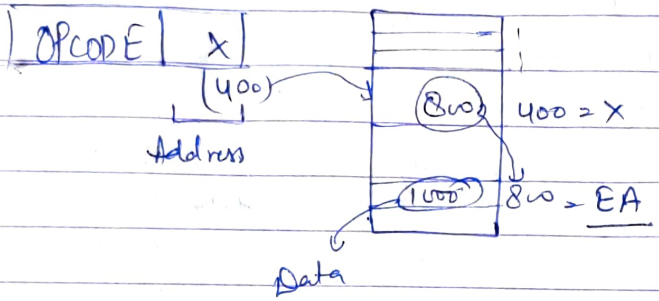
EA = Add. Part of instruction + content of CPU register

(7) Indirect 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.

It is used to implement pointers & passing parameters.



ADD x
AC ← AC + M[M[x]]

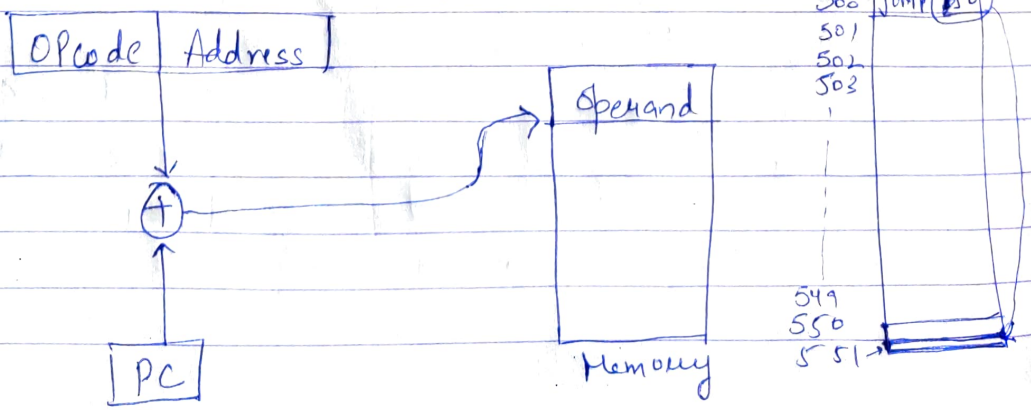
EA = M[x]

Here computation is increased, to calculate the operand.

(8) Relative Addressing Mode :

Effective Address = PC + offset

* Displacement = Original − 1.



Why use Offset ?

The no. of bits required for 1024 words will be 10 bits.

⇒ 10 bits to represent address & the instruction set size will also increase.

⇒ So, to decrease the size of instruction set, we will use Offset.

Relative Addressing mode explaination :

The content of PC is added to the address part of the instruction in order to obtain the effective address.

# Ex: PC = 825
Offset = 24

The instruction at location 825 is read from memory during the fetch phase & the PC is then incremented by one to 826.

Effective address = 826 + 24 = 850.

This is 24 memory locations forward from the address of the next instruction.

It is often used with branch-type instructions.

(9) Indexed Addressing Mode :

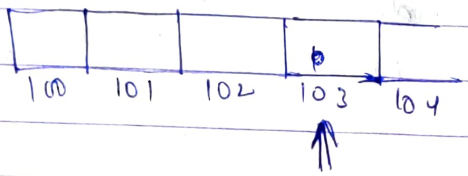
Effective address : Base register + Index register content

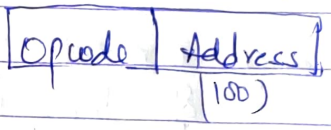
The Index register is a special CPU register that contains an index value.

It is used to access or implement array efficiently

Multiple registers are required to implement.

Any element can be accessed without changing instruction.





Index Register
[3]

EA = 100 + 3
= 103

The address field of the instruction defines the beginning address of a data array in memory.

Each operand in the array is stored in memory relative to the beginning address. The distance between the beginning address & the address of the operand is the index value stored in the Index register.

Any operand in the array can be accessed with the same instruction provided that the Index Register contains the correct Index value.

10. Base Register Addressing Mode:

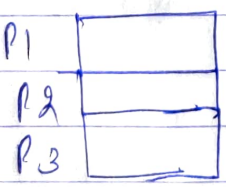
Effective address : Base register + Address
{Displacement}

A base register is assumed to hold a base address & the address field of the instruction gives a displacement relative to this base address.

It is used in computers to facilitate the relocation of programs in memory.

When programs & data are moved from one segment of memory to another, as required in multiprogramming systems, the address values of instructions must reflect this change of position.

With a base register, the displacement values of instructions do not have to change. Only the value of the base register requires updating to reflect the beginning of a new memory segment.



~~Addr~~ Base register + Displacement

150	60

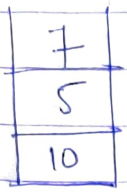
↑ Can be changed.

↳ Do not change.

11. Stack Addressing Mode :

In this addressing mode,

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



Eg: ADD

↳ This instruction simply pops out two symbols contained at the top of the stack.

↳ The addition of those two operand is performed.

↳ The result so obtained after addition is pushed again at the top of the stack.

↳ Categorisation based on Instruction set

CISC	RISC
# Complex instruction set Computer	# Reduced Instruction set Computer
Large no. of instructions	• Less No. of instructions
Variable length instruction format	• Fixed length instruction format.
Large no. of addressing modes support.	• Supports few no. of addressing modes.
Cost is high.	• Less cost.
More powerful.	• Less powerful.
Several cycle instructions.	• Single cycle instructions.
Manipulation directly in memory	• Only in registers.
Microprogrammed control unit.	• Hardwired control unit.
Ex: Mainframes, Motorola 6800, Intel 8080.	• Fugaku, MIPS, ARM, SPARC.
MULT 2:2, 3:3 X, Y [x = x*y]	LOAD A, 2:2 LOAD B, 3:3 PROD A, B STORE 2:2, A.