Instruction format

1.Three address instruction format

2.Two address instruction format

3.One address instruction format

4. Zero address instruction format

Instruction

- Programme: a sequence of instructions
- Instructions: An instruction is a command given to the computer to perform a specific operation.
- An instruction is a group of bits (binary code) that instructs the computer to perform a specific operation.
- The purpose of an instruction is to specify both operation to be performed by CPU and the set of a operands or data.
- Operation. Include add, subtract, multiplication, shift etc. Operations are performed on data (operands).
- Operands include the input data and the results that are produced.

CPU organisations and types of instructions

- Computers may have instructions of several different lengths containing varying number of addresses (address field maybe 3,2,1,0).
- The number of address field in the instruction format depends on the internal organisation of its registers.
- Most computers fall into one of the three types of CPU organisations
- Single accumulator organisation
- General register organisation
- Stack organisation

Types of instruction format/ types of addres instructions

- 1.General register organisation
- Three address instruction format
- Two address instruction format
- 2.Single accumulator organisation
- One address instruction format
- 3.Stack Organisation
- Zero address instruction format

➔ To illustrate the influence of the number of addresses on computer programs, we will evaluate the arithmetic statement

$$X = (A + B) \cdot (C + D)$$

using zero, one, two, or three address instructions. We will use the symbols ADD, SUB, MUL, DIV for the four arithmetic operations; MOV for the transfer-type operation; and LOAD STORE for transfers to and from memory and AC register. We will assume that the operands in memory addresses A, B, C, and D, and the result must be stored in memory at address X.

Three-Address Instructions:

```
ADD   R1, A, B    R1←M[A] + M[B]
ADD   R2, C, D    R2←M[C] + M[D]
MUL   X, R1, R2   M[X]←R1*R2
```

Two-Address Instructions:

```
MOV   R1, A     R1←M[A]
ADD   R1, B     R1←R1 + M[B]
MOV   R2, C     R2←M[C]
ADD   R2, D     R2←R2 + M[D]
MUL   R1,R2     R1←R1*R2
MOV   X, R1     M[X]←R1
```

One-Address Instructions:

```
LOAD    A    AC←M[A]
ADD     B    AC←AC + M[B]
STORE   T    M[T]←AC
LOAD    C    AC←M[C]
ADD     D    AC←AC + M[D]
MUL     T    AC←AC*M[T]
STORE   X    M[X]←AC
```

Zero-Address Instructions:

```
PUSH   A   TOS←A
PUSH   B   TOS←B
ADD        TOS←(A + B)
PUSH   C   TOS←C
PUSH   D   TOS←D
ADD        TOS←(C + D)
MUL        TOS←(C + D)*(A + B)
POP    X   M[X]←TOS
```

RISC Instructions:

```
LOAD    R1, A          R1←M[A]
LOAD    R2, B          R2←M[B]
LOAD    R3, C          R3←M[C]
LOAD    R4, D          R4←M[D]
ADD     R1, R1, R2     R1←R1 + R2
ADD     R3, R3, R2     R3←R3 + R4
MUL     R1, R1, R3     R1←R1*R3
STORE   X, R1          M[X]←R1
```