Types of Instruction Formats

1. Zero Address Instruction

This instruction does not have an operand field, and the location of operands is implicitly represented. The stack-organized computer system supports these instructions. To evaluate the arithmetic expression, it is required to convert it into reverse polish notation.

Example: Consider the below operations, which shows how
X = (A + B) ∗ (C + D) expression will be written for a stack-organized computer.

TOS: Top of the Stack

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**

2. One Address Instruction

This instruction uses an implied accumulator for data manipulation operations. An accumulator is a register used by the CPU to perform logical operations. In one address instruction, the accumulator is implied, and hence, it does not require an explicit reference. For multiplication and division, there is a need for a second register. However, here we will neglect the second register and assume that the accumulator contains the result of all the operations.

Example: The program to evaluate X = (A + B) ∗ (C + D) is as follows:

LOAD	**A**		**AC ← M [A]**
ADD	**B**		**AC ← A [C] + M [B]**
STORE		**T**	**M [T] ← AC**
LOAD	**C**		**AC ← M [C]**

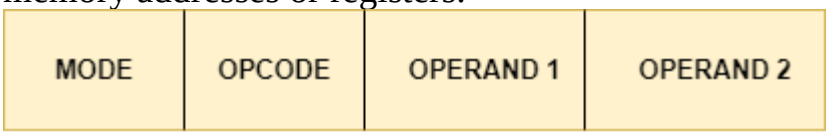
```
ADD     D       AC ← AC + M [D]
MUL     T       AC ← AC * M [T]
STORE   X       M [X] ← AC
```

All operations are done between the accumulator(AC) register and a memory operand.
M[ ] is any memory location.
M[T] addresses a temporary memory location for storing the intermediate result.
This instruction format has only one operand field. This address field uses two special instructions to perform data transfer, namely:

- LOAD: This is used to transfer the data to the accumulator.
- STORE: This is used to move the data from the accumulator to the memory.

3. Two Address Instructions

This instruction is most commonly used in commercial computers. This address instruction format has three operand fields. The two address fields can either be memory addresses or registers.



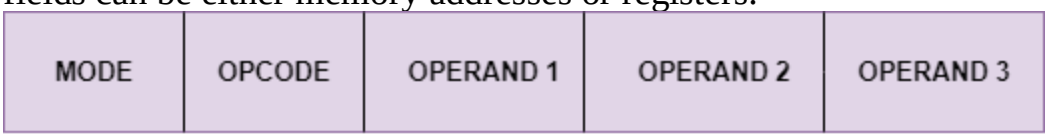
Example: The program to evaluate X = (A + B) * (C + D) is as follows:

```
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
```

The MOV instruction transfers the operands to the memory from the processor registers. R1, R2 registers.

4. Three Address Instruction

The format of a three address instruction requires three operand fields. These three fields can be either memory addresses or registers.



Example: The program in assembly language X = (A + B) ∗ (C + D) Consider the instructions given below that explain each instruction's register transfer operation.

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 processor registers, R1 and R2.
The symbol M [A] denotes the operand at memory address symbolized by A. The operand1 and operand2 contain the data or address that the CPU will operate. Operand 3 contains the result's address.

Advantages and Disadvantages
Here are some advantages and disadvantages of each instruction format:

1. **Single accumulator organization:**
   Advantages: Simple design, low memory requirements, efficient for certain types of computations.
   Disadvantages: Limited parallelism, limited functionality.

2. **General register organization:**
   Advantages: More versatile, supports parallel processing, more efficient for certain types of computations. Disadvantages: More complex design, higher memory requirements.

3. **Stack organization:**
   Advantages: Simple design, low memory requirements, supports recursive function calls.
   Disadvantages: Limited parallelism, slow access to non-top elements.