INSRTRUCTION FORMAT

LECTURER :Dalia Adil

The Accumulator

The accumulator in a CPU is a special register used to store intermediate results of arithmetic and logic operations. It plays a key role in processing instructions, as many operations in assembly language directly involve the accumulator.

Here are a few key points about the accumulator

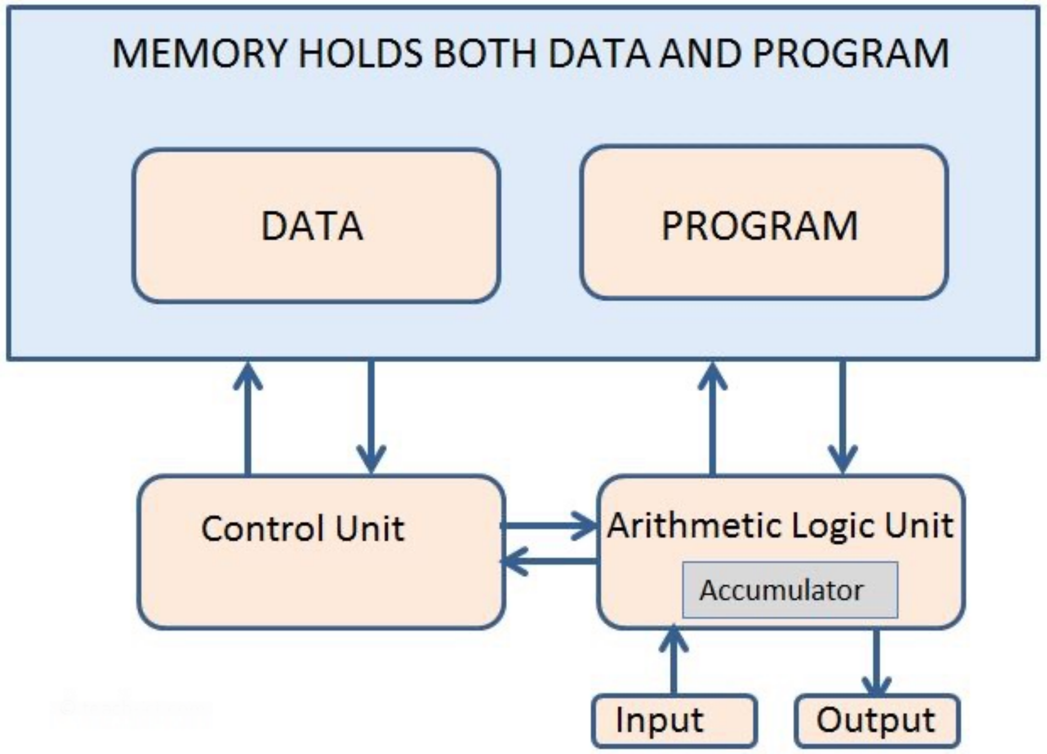
1.Intermediate Storage: It temporarily holds data during calculations. For example, when adding two numbers, one number might be loaded into the accumulator, and then the second number is added to it.

- For example, in the operation "3 + 4 + 5," the accumulator would hold the value 3, then the value 7, then the value 12. The benefit of an accumulator is that it does not need to be explicitly referenced, which conserves data in the operation statement.

2.Speed: Accessing the accumulator is typically faster than accessing other types of memory, making operations more efficient.

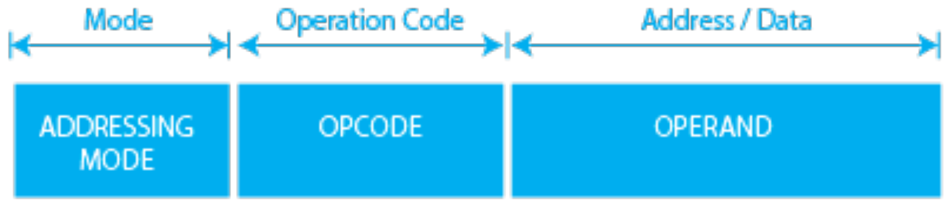
3. Operation: Many CPU instructions are designed to operate on the accumulator directly, simplifying the instruction set and speeding up execution.

4. Historical Context: In early computers, the accumulator was a primary register, and many architectures still maintain its use, even if they have additional registers.



Instruction Format in Computer Architecture?

- **Addressing Mode:** The addressing mode indicates how the data is represented.
- **Opcode:** The opcode part indicates the operation type on the data.
- **Operand:** The operand part indicates either the data or the address of the data.



Example of Instruction format

The three most common cpu organization:

1.Single Accumulator Organization

- An implied accumulator register is used in every action on a system. This type of computer utilizes one address field for the instruction format.
- For instance, the assembly language command ‘ADD’ defines the instruction for arithmetic addition.
  The action is produced by the ADD instruction, where X is the address of the operand.
- $AC \leftarrow AC + M[X]$.
- AC is the accumulator register, and M[X] symbolizes the memory word located at address X.

2. General Register Organization

- In their command format, general register-type computers use two or three address fields. Each address column identifies a memory or a processor register. The procedure R1 R + M [X] is specified by an instruction with the symbol ADD R1, X.
  The memory address X and register R1 are the two address elements for this instruction.

3. Stack Organization

- The PUSH and POP commands on a computer with a stack organization need an address field. As a result, the word at address X is pushed to the head of the stack by the instruction PUSH X. The stack pointer immediately updates. Since the operation is done on the top two items of the stack, stack-organized computers don’t need an address field for the operation type instructions.

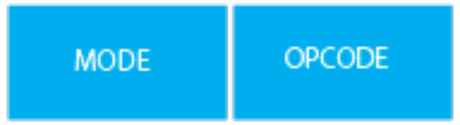
Types of Instruction Format in Computer Architecture

1. Zero Address Instruction

- Can be found in astack –organized computer. Example of Zero address instruction : Consider the actions below, which demonstrate how the expression X = (A + B) (C + D) will be formatted 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
```

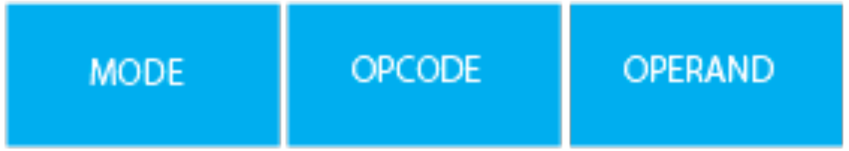


2. One Address Instruction

- Use an implied ac register for all data manipulation

Example of One address instruction**:** 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
```



LOAD: This is used to transfer the data to the accumulator.

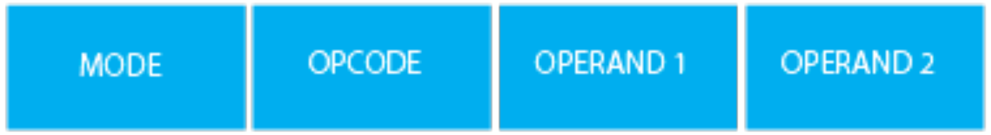
STORE: This is used to move the data from the accumulator to the memory.

3. Two Address Instructions

The majority of commercial computers use this command. There are three operand fields in this address command format. Registers or memory addresses can be used in the two address sections.

Example of Two address instruction: 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 command moves the operands from the processor registers to the memory. sensors R1, R2.

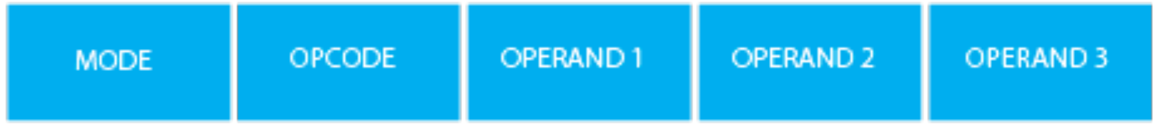
4. Three Address Instruction

Example of Three address instruction: The assembly language program X = (A + B) * (C + D) Take a look at the instructions that follow, which describe the register transfer procedure for each instruction.

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

R1 and R2 are the two CPU registers.

The operand at the memory location represented by A is indicated by the symbol M [A]. The data or location that the CPU will use is contained in operands 1 and 2. The address of the output is in operand 3.


MODE
OPCODE
OPERAND 1
OPERAND 2
OPERAND 3