CPU

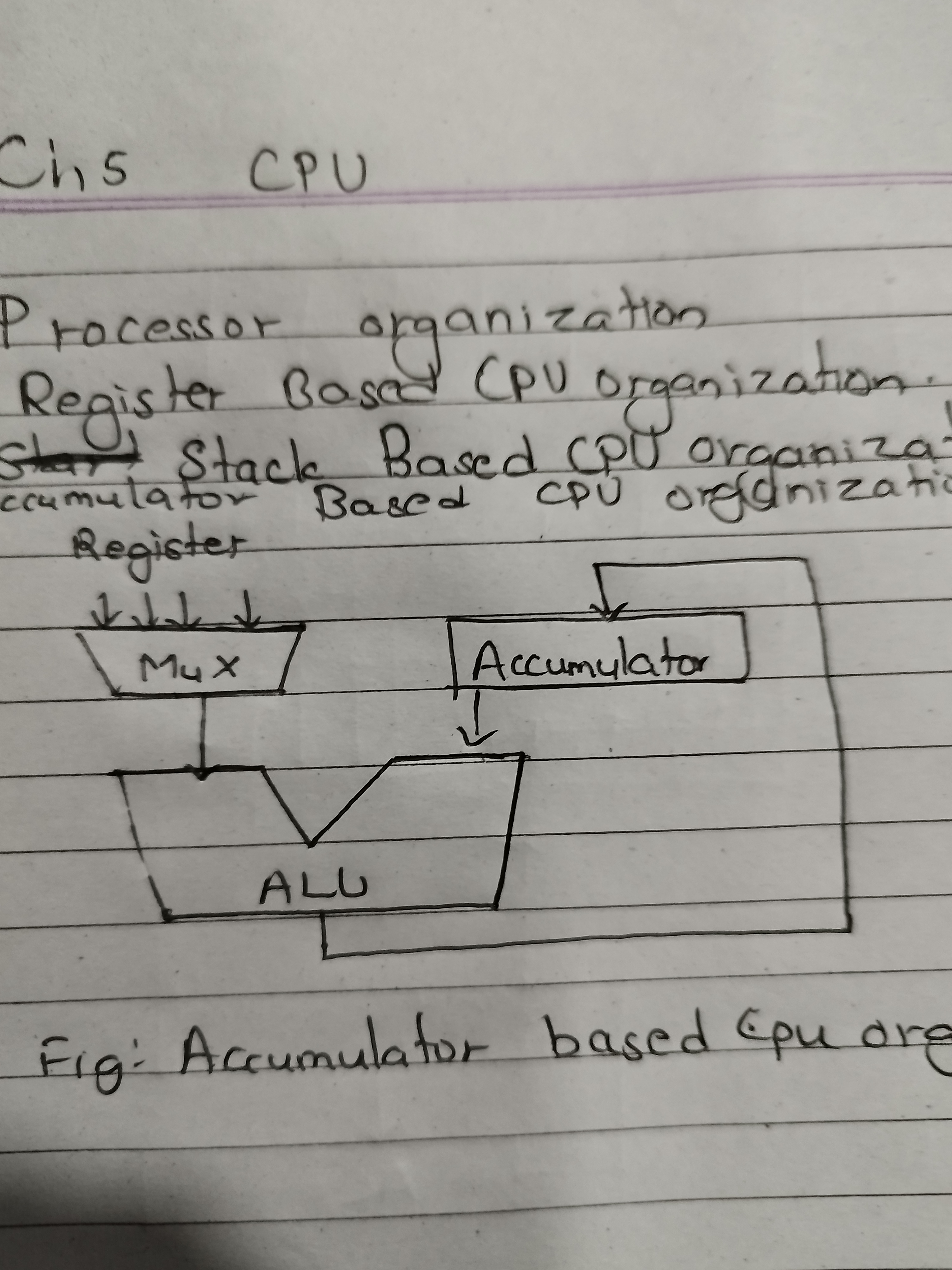
Processor Organization:

1. Register based CPU organization

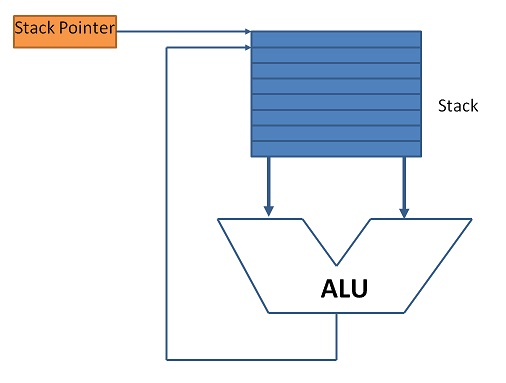
2. Accumulator based CPU organization

3. Stack based CPU organization

**Accumulator based CPU organization:**



**Stack Based CPU organization:**

****

**Instruction Format:**

1. Three address instruction format

→ MUL R1 ,A ,B // R1 ← M[A] \* M[B]

2. Two address instruction format

→ MOV R1 ,A // R1← M[A]

→ MUL R1 ,B // R1 ← R1 \* M[B]

3. One address instruction format

→ MUL R1 // AC ← AC \* R1

4. Zero address instruction format [stack based CPU organization]

→ The expression (A\*B)+(C\*D) in zero address format will be as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | D |  |  |
|  | B |  | C | C | C\*D |  |
| A | A | A\*B | A\*B | A\*B | A\*B | (A\*B)+(C\*D) |

\*each column is a step and step progresses from left to right.

Program:

PUSH A

PUSH B

MUL

PUSH C

PUSH D

MUL

ADD

POP X

**\*** to write this program convert expression to postfix notation.

**Addressing Modes:**

The way of specifing operand in an instruction.

Instruction = opcode + operand

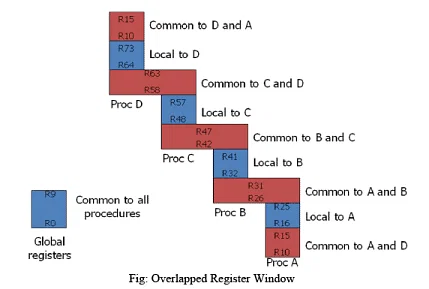
* Immediate → MVI A , 05H
* Implied → CMA
* Direct → LDA 2000H
* Indirect → ILDA 1000H (Address of address of operand)
* Register → ADD B
* Register Indirect → ADD M (M contains address of operand)
* Displacement
  + Relative → EA = address + PC (EA=effective address)
  + Index → EA = address + IR (IR=index register)
  + Base → EA = address + CS (CS=code segment)
  + Base + Index → EA = address + CS + DI (DS=destination index)
* Auto Increment / Decrement

**RISC vs CISC:**

|  |  |
| --- | --- |
| **Reduced Instruction Set Computer** | **Complex Instruction Set Computer** |
| Small number of instructions | Large number of instructions |
| Simple hardware | Complex hardware |
| Complex software/complier | Simple software/compiler |
| Large number of registers | Less number of registers |
| Less addressing mode | Large number of addressing mode |
| Hardwired control unit | Microprogrammed control unit |
| Load and store instructions are used for memory access | Multiple instructions available for momery access |
| Instruction completes in one cycle | Instructions completes in multiple cycles |
| Efficient pipeline implimentation | Pipeline if implimented not efficient |
| Fixed length instruction | Variable length instruction |
| **Eg ARM , RISC-V , etc** | **Eg VAX , x86\_64 , etc** |

**Register overlapped window:**

Available in RISC having large number of registers.



Number of gobal registers = G

Number of local registers in each window = L

Number of common registers common to windows = C

Number of windows = W

Window size = L + 2C + G

Register file = (L+C)W + G