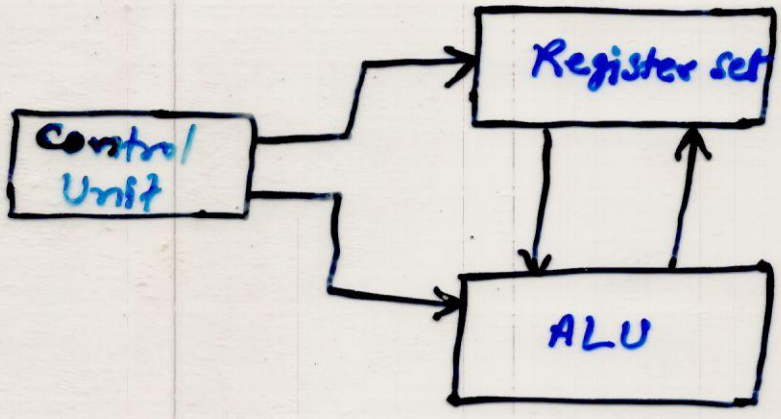
Central Processing Unit

The part of the computer that performs the bulk of data processing operations is called the central processing unit (CPU).

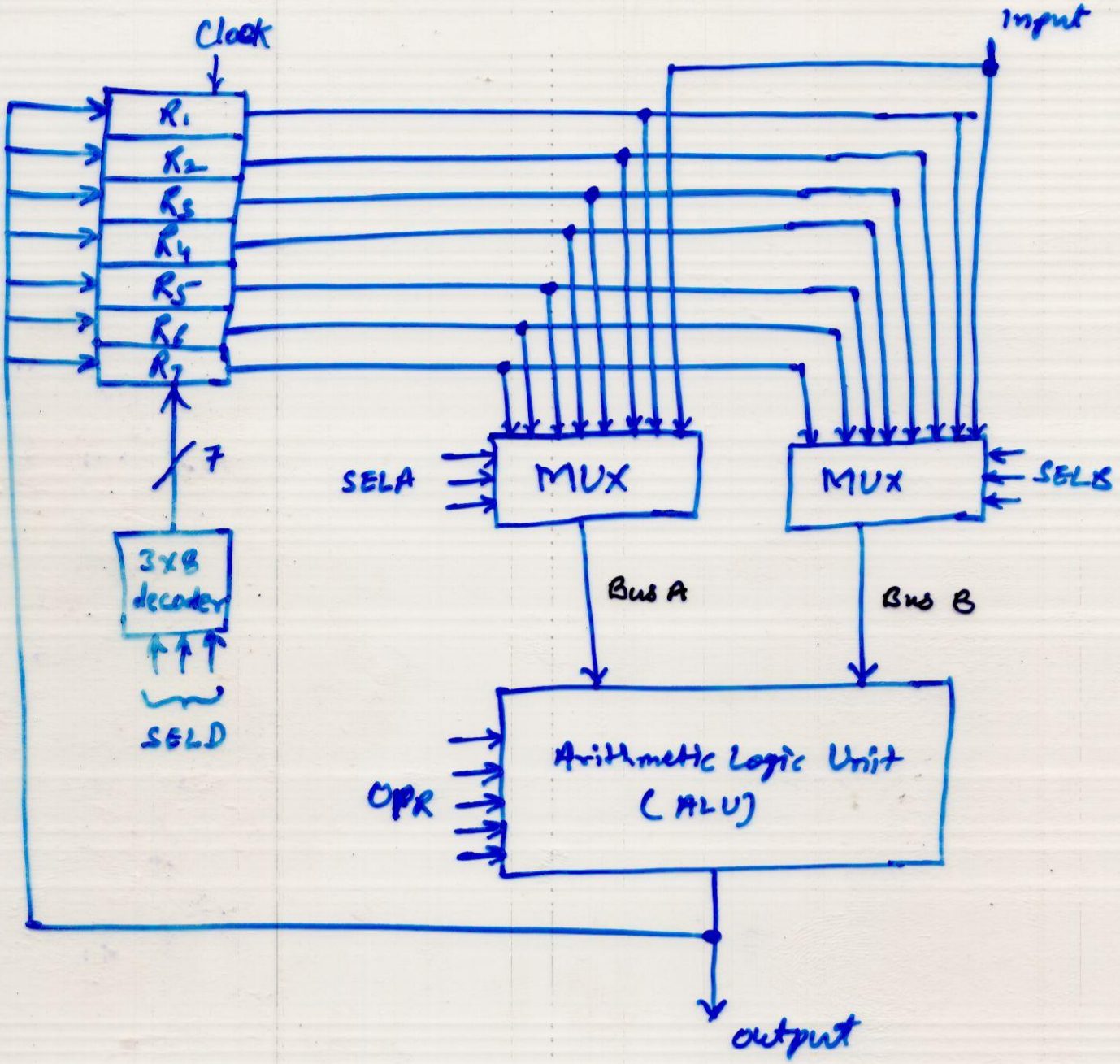
The major components of CPU are following:

1. ALU (Arithmetic Logic Unit)
2. Register set
3. Control Unit

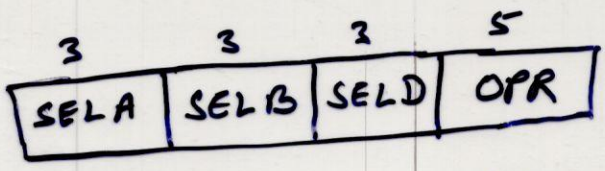


- The register set stores intermediate data used during the execution of the instructions.
- The arithmetic logic unit performs the required microoperations for executing the instructions
- The control unit supervise the transfer of information among the

General Register Organization



- The selection lines in each multiplexers (MUX) → select one register or input data for the particular bus.
- The register that receives the information from the output bus is selected by a decoder.



(Control word)

- The output of each register is connected to two multiplexers to form the two buses A and B.
- The register that recieves the information from the output bus is selected by a decoder.

Example:

$$R_1 \leftarrow R_2 + R_3$$

1. MUX A selector (SELA): to place the content of R_2 into bus A.
2. MUX B selector (SELB): to place to content of R_3 into bus B.
3. ALU operation selector (OPR): to provide the arithmetic addition A + B
4. Decoder destination selector (SELD): to transfer the content of the output bus into R_1.

Types of CPU Organizations

1. Single accumulator organization
2. General register Organization
3. Stack organization

All operations in ~~a~~ accumulator-type organization are performed with an implied accumulator register. The instruction format in this type of computer uses one address field.

e.g. ADD X (means $AC \leftarrow AC + M[X]$)

The instruction format in a general register type of organization needs three register address fields.

e.g.

ADD R_1, R_2, R_3 $\{ R_1 \leftarrow R_2 + R_3 \}$

or

ADD R_1, R_2 {Two address fields $R_1 \leftarrow R_1 + R_2$ }

The stack organization have PUSH and POP instruction which require one address field.

PUSH X

but operation-type instructions do not need an address field.

Three-Address Instruction

Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.

The program in assembly language that evaluates $X = (A+B) * (C+D)$ is:

ADD R_1, A, B	$R_1 \leftarrow M[A] + M[B]$
ADD R_2, C, D	$R_2 \leftarrow M[C] + M[D]$
MUL X, R_1, R_2	$M[X] \leftarrow R_1 * R_2$

Two-Address Instruction

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

MOV R_1, A	$R_1 \leftarrow M[A]$
ADD R_1, B	$R_1 \leftarrow R_1 + M[B]$
MOV R_2, C	$R_2 \leftarrow M[C]$
ADD R_2, D	$R_2 \leftarrow R_2 + M[D]$
MUL R_1, R_2	$R_1 \leftarrow R_1 * R_2$
MOV X, R_1	$M[X] \leftarrow R_1$

One-Address Instruction

One-address instruction use an implied accumulator (AC) register for all data manipulation. The program to evaluate $X = (A+B) * (C+D)$ is:

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

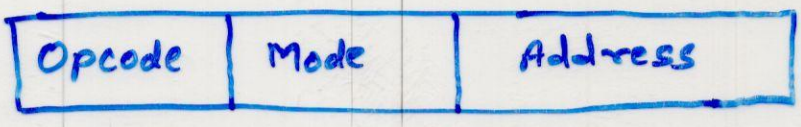
Zero-Address instruction

The program to evaluate $X = (A+B) * (C+D)$ IS

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

Addressing Mode

The addressing mode specifies the rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. Instruction format with mode field is shown below.



Different Addressing mode

1. **Implied Mode:-** In this mode the operands are specified implicity in the definition of the instruction (zero addressing field).

   e.g. - complement of accumulator.

2. **Immediate Mode:-** In this mode the operand is specified in the instruction itself. In other words, An immediate-mode instruction has an Operand field rather than an address field.

3. **Register Mode:-** In this mode the operands are in registers that reside within the CPU.

Register Indirect Mode:- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.

Direct Address mode:- In this mode the effective address is equal to the address part of the instruction.

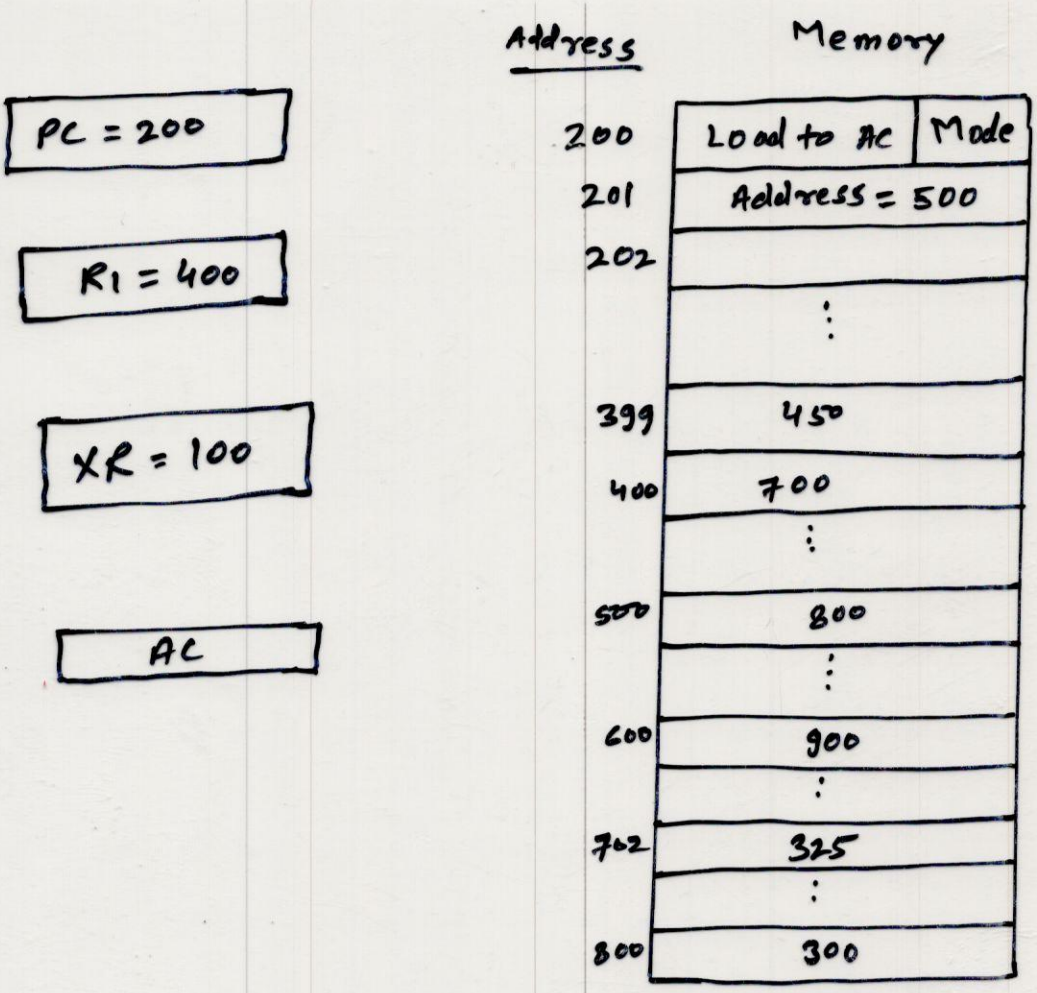
Indirect Address Mode:- In this mode the address field of the instruction gives the address where the effective address is stored in memory.

Relative Address modes:- In this mode the content of the program counter is ~~address~~ added to the address part of the instruction in order to obtain the effective address.

Indexed Address mode:- In this mode the content of an index register is added to the the address part of the instruction to obtain the effective address.

Base Register Addressing Mode:- In this mode the content of a base register is added to the address part of the

Numerical Example



Tabular List of Numerical example

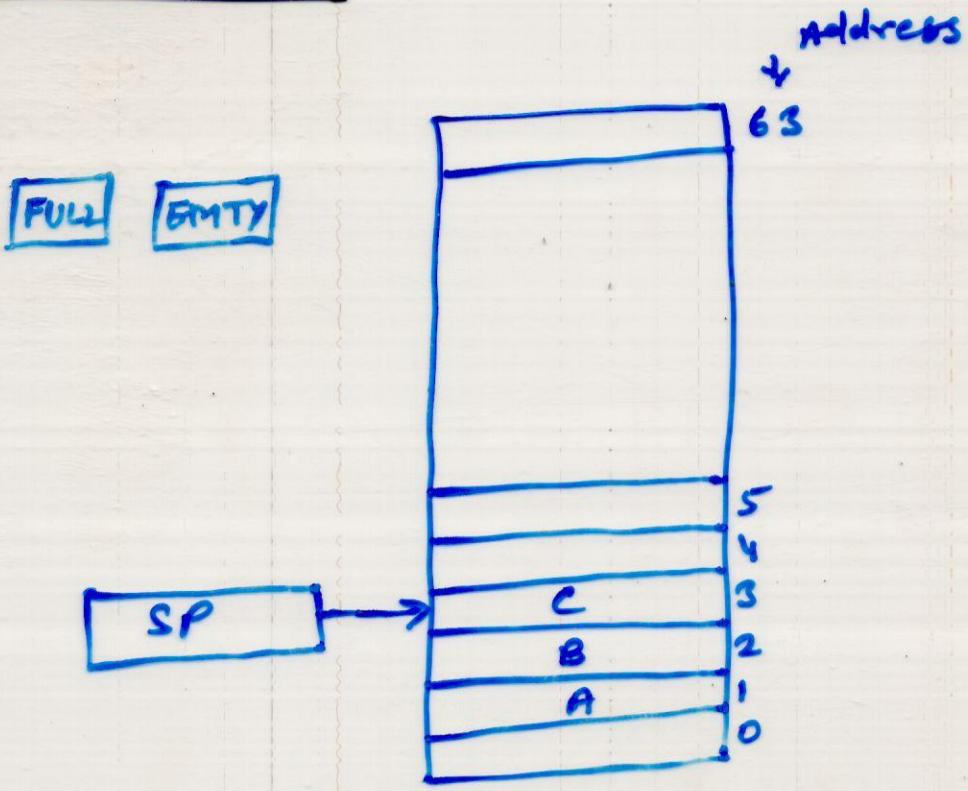
	Addressing Mode	Effective address	content of AC
1.	Direct address	500	800
2.	Immediate operand	201	500
3.	Indirect address	800	300
4.	Relative address	702	325
5.	Indexed address	600	900
6.	Register	—	400
7.	Register indirect	400	700

Stack Organization

A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved (LIFO).

- The register that holds the address for the stack is called a stack pointer (SP). Its value always points at the top item in the stack.
- The two operation of a stalk:
  1. push (insertion)
  2. pop (deletion)

Register Stack



Initially

SP = 0

EMTY = 1

FULL = 0