Pamantasan ng
Lungsod ng Maynila

‘72

Simple as Possible
Computer - 1

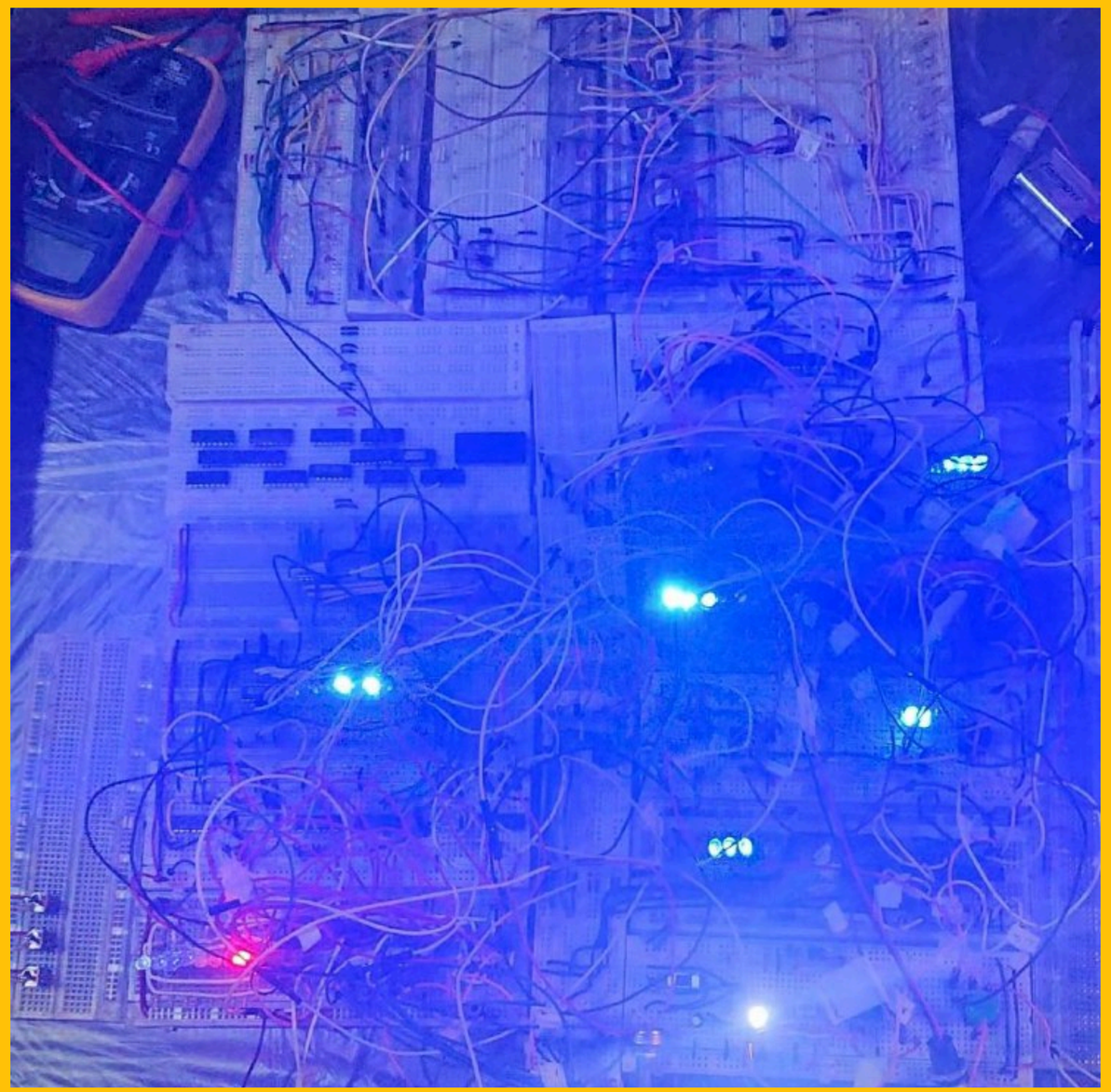
Group Numbers:

1 - Apelledo’s Group
3 - Goyena’s Group

Professor:

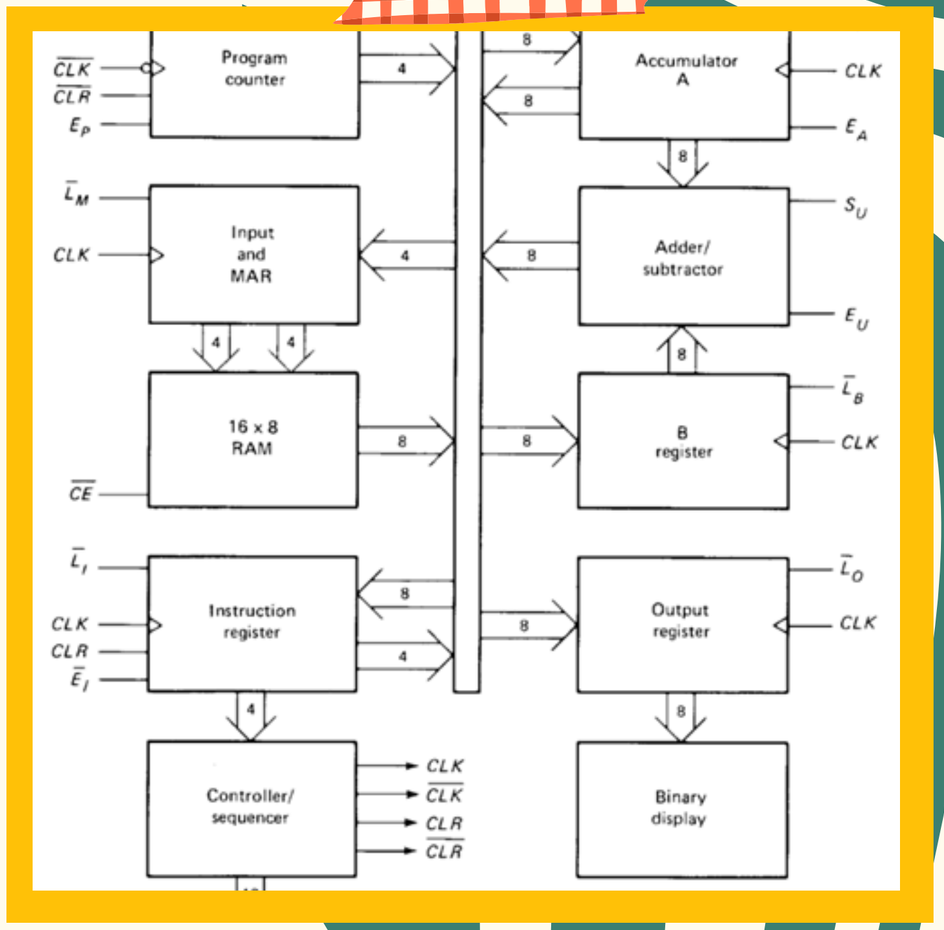
Elsa S. Pascual

SAP - 1



SAP - 1

- SAP-1 is a beginner-friendly computer model.
- It teaches basic concepts of how computers work.
- Designed to be simple, avoiding complex details.
- It is the first stage in the evolution toward modern computers.
- Helps you learn architecture, programming, and circuits.
- Mastering SAP-1 prepares you for more advanced computers.



Clock

What it is:

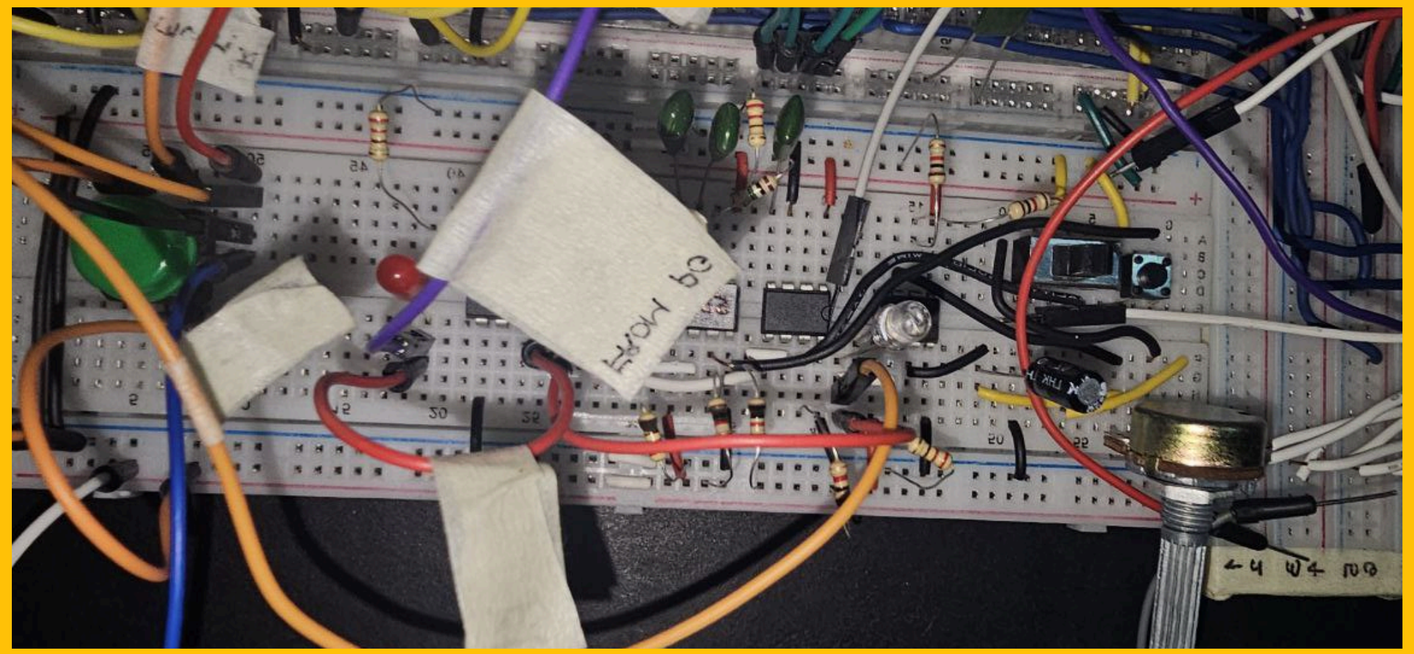
Generates a series of **timing pulses** used to synchronize and control the various steps in the instruction cycle.

Why it’s needed:

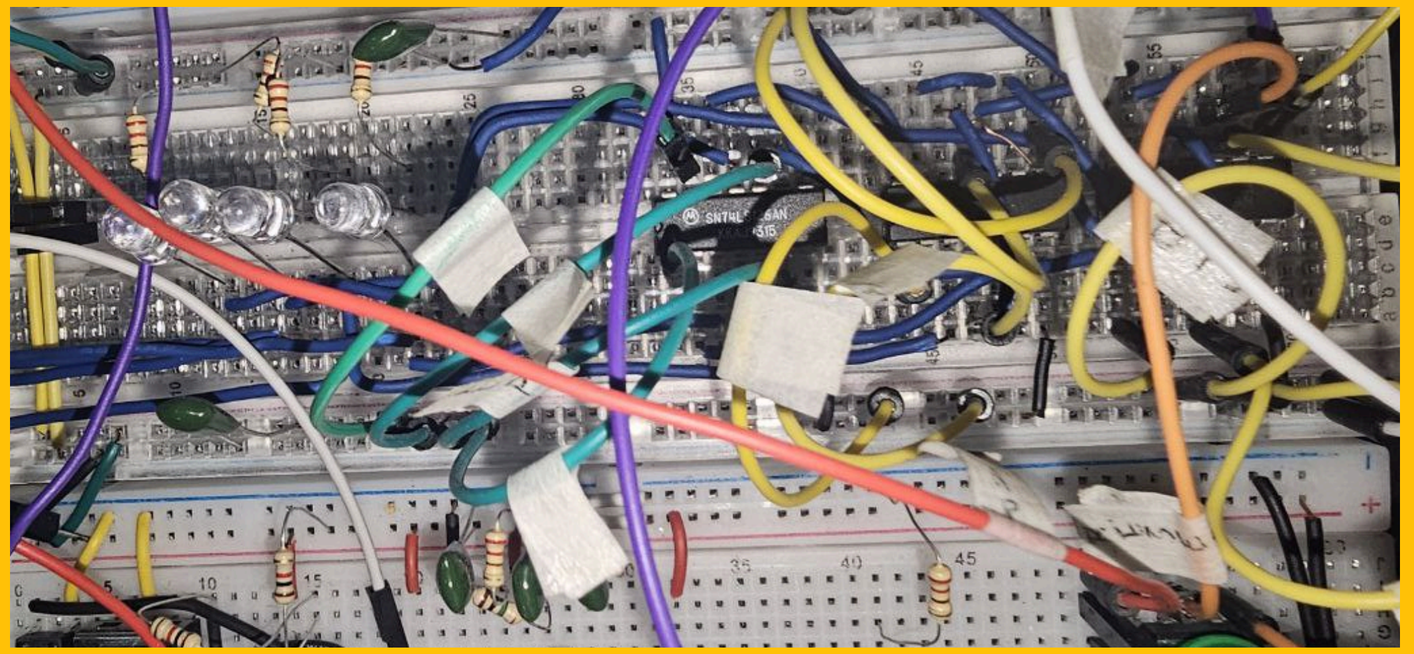
Ensures that each part of the SAP-1 processor works in sync.

How it works:

1. Produces periodic timing pulses that initiate each stage of the instruction cycle.
2. These pulses synchronize the operations of the program counter, ALU (adder-subtractor), registers, and memory to execute instructions sequentially.



Program Counter



What it is:

A register that stores the memory address of the next instruction to be fetched.

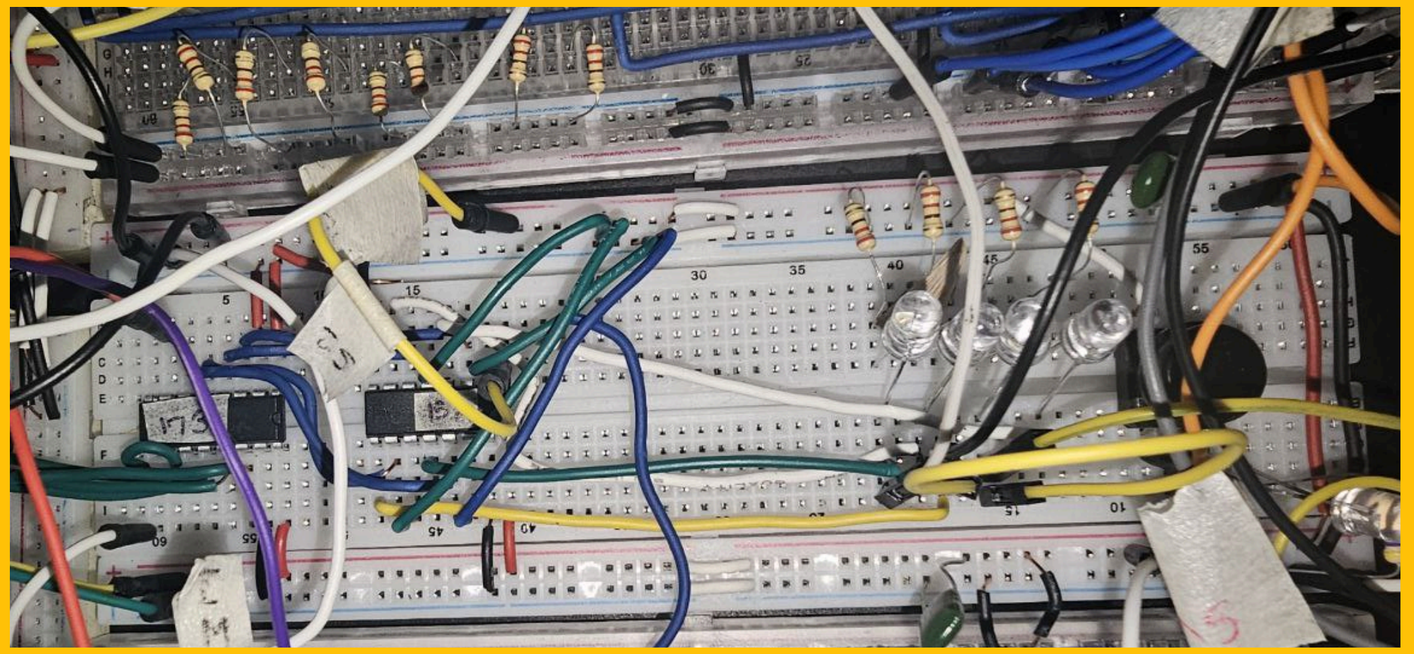
Why it's needed:

- Ensures the CPU executes instructions in the correct sequence.
- Automatically increments after each instruction fetch, moving to the next instruction in memory.

How it works:

1. At the start of execution, the PC is set to the address of the first instruction.
2. The address in PC is sent to the Memory Address Register (MAR).
3. After fetching the instruction, PC increases by 1 to point to the next instruction.

Multiplexer (MUX) and MemoryAddress Register (MAR)



What they are:

- MUX (Multiplexer): A switch that selects which address to send to the MAR (either from the PC or a manual input).
- MAR (Memory Address Register): Holds the address of the memory location being accessed.

Why they're needed:

- MAR ensures the correct memory location is accessed for fetching instructions or data.
- MUX allows manual addressing, useful for debugging or loading programs manually.

How they work:

1. The PC outputs an address.
2. MUX selects the PC's address (or manual input).
3. The selected address is sent to the MAR.
4. MAR holds the address while the memory fetches the data stored at that location.

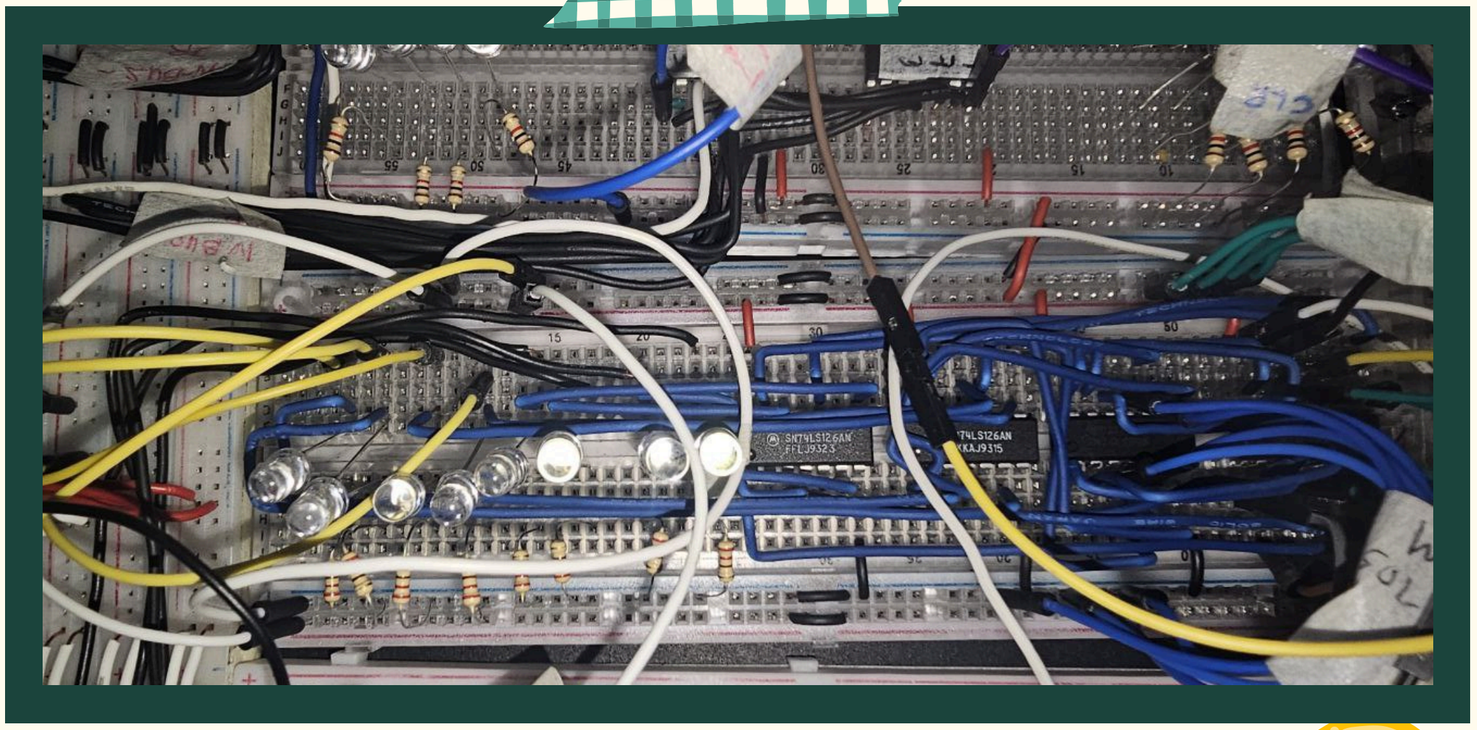
RAM (Random Access Memory)

What it is:

- Stores both program instructions and data.
- Typically consists of 16 memory locations, each storing an 8-bit instruction or data.

Why it’s needed:

- Acts as the short-term memory of the computer, holding everything needed for execution.
- The CPU retrieves instructions and data from RAM during operation.



How it works:

1. The MAR provides the address of the instruction or data.
2. RAM outputs the value stored at that address.
3. The instruction/data is sent through the W Bus to the CPU.

Instruction Register (IR)

What it is:

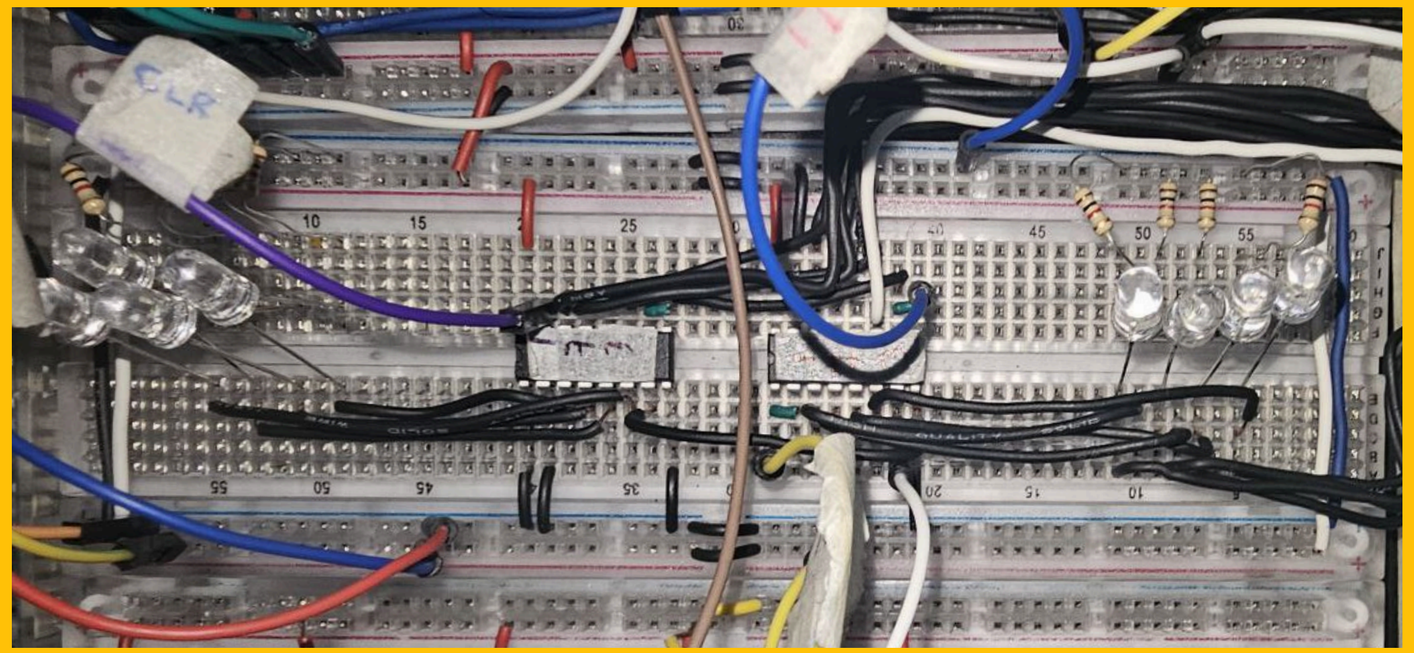
- A register that stores the current instruction being executed.
- Holds both the opcode and the operand (memory address or data).

Why it’s needed:

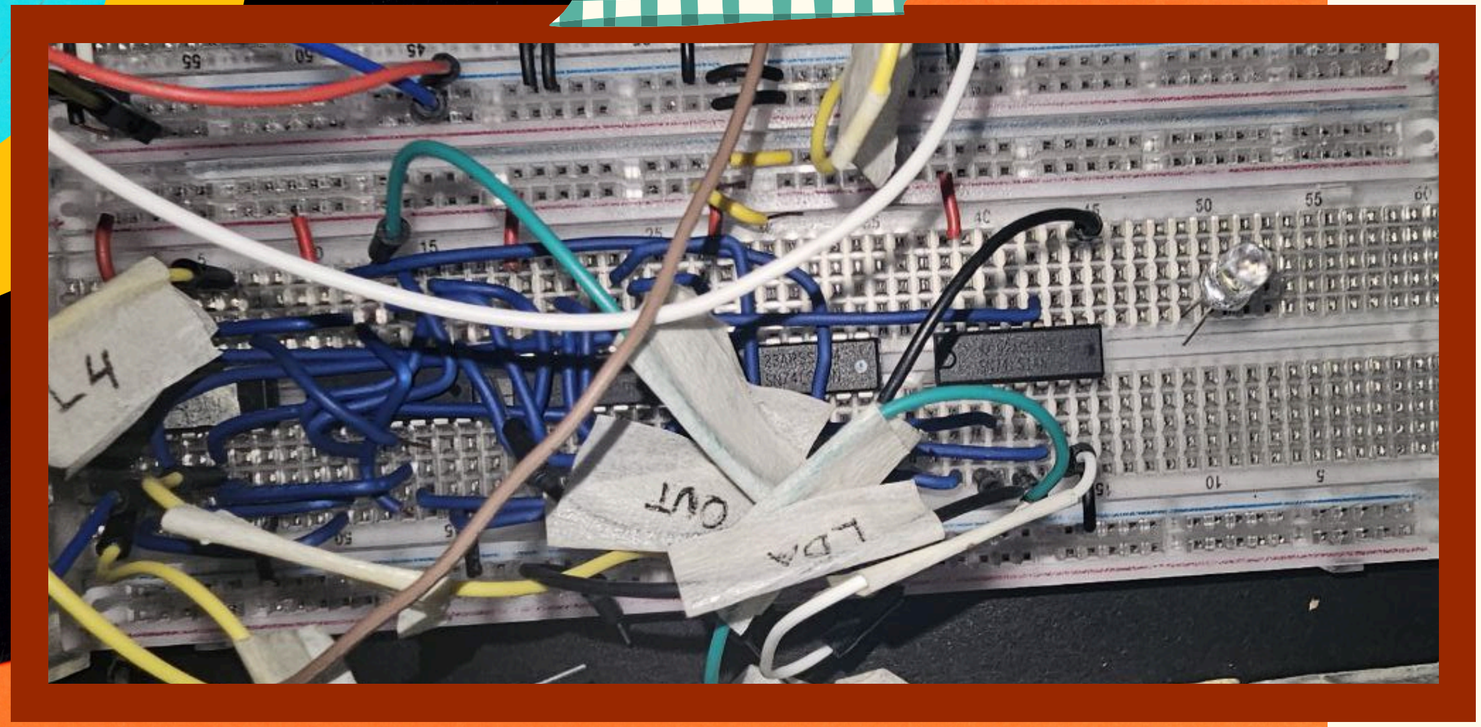
- Keeps the fetched instruction ready for processing.
- Allows the Instruction Decoder to interpret and execute it.

How it works:

1. The instruction is fetched from RAM.
2. It is stored in the IR.
3. The decoder reads the opcode and sends control signals accordingly.



Instruction Decoder



What it is:

- A circuit that interprets the opcode (operation code) of an instruction.
- Determines what action the CPU must perform.

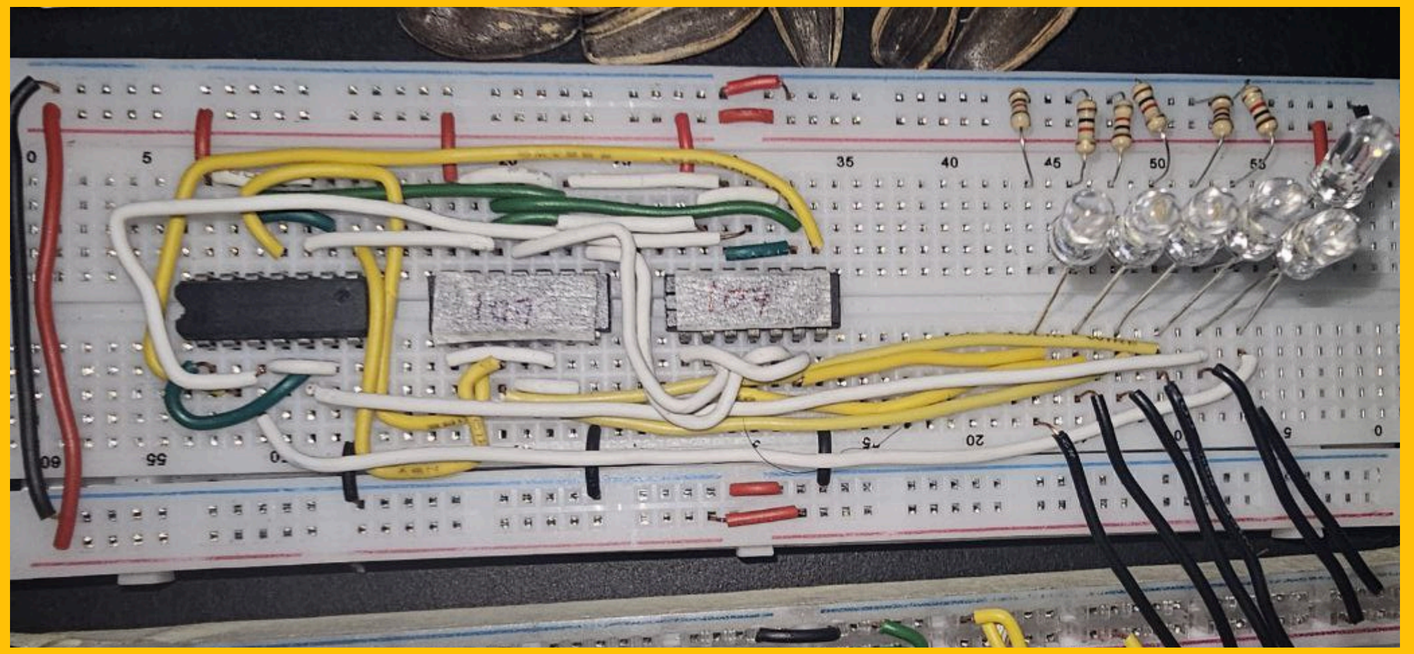
Why it's needed:

- Translates binary instructions into specific control signals for the CPU components.
- Ensures each instruction is properly executed by activating the right parts of the computer.

How it works:

1. The Instruction Register (IR) holds the instruction.
2. The decoder extracts the opcode (first 4 bits).
3. Based on the opcode, it generates control signals to execute the corresponding operation (e.g., load, add, store).

Ring Counter



What it is:

- A special counter that cycles through a fixed number of states.
- Provides timing signals for each step of instruction execution.

Why it’s needed:

- Ensures the correct sequence of operations (fetch, decode, execute).
- Works like a step-by-step guide for the computer.

How it works:

1. The counter moves through states T1, T2, T3, T4, etc.
2. Each state corresponds to a part of the instruction cycle (fetch, decode, execute).
3. Resets after a full cycle, ready for the next instruction.

Control Matrix

What they are:

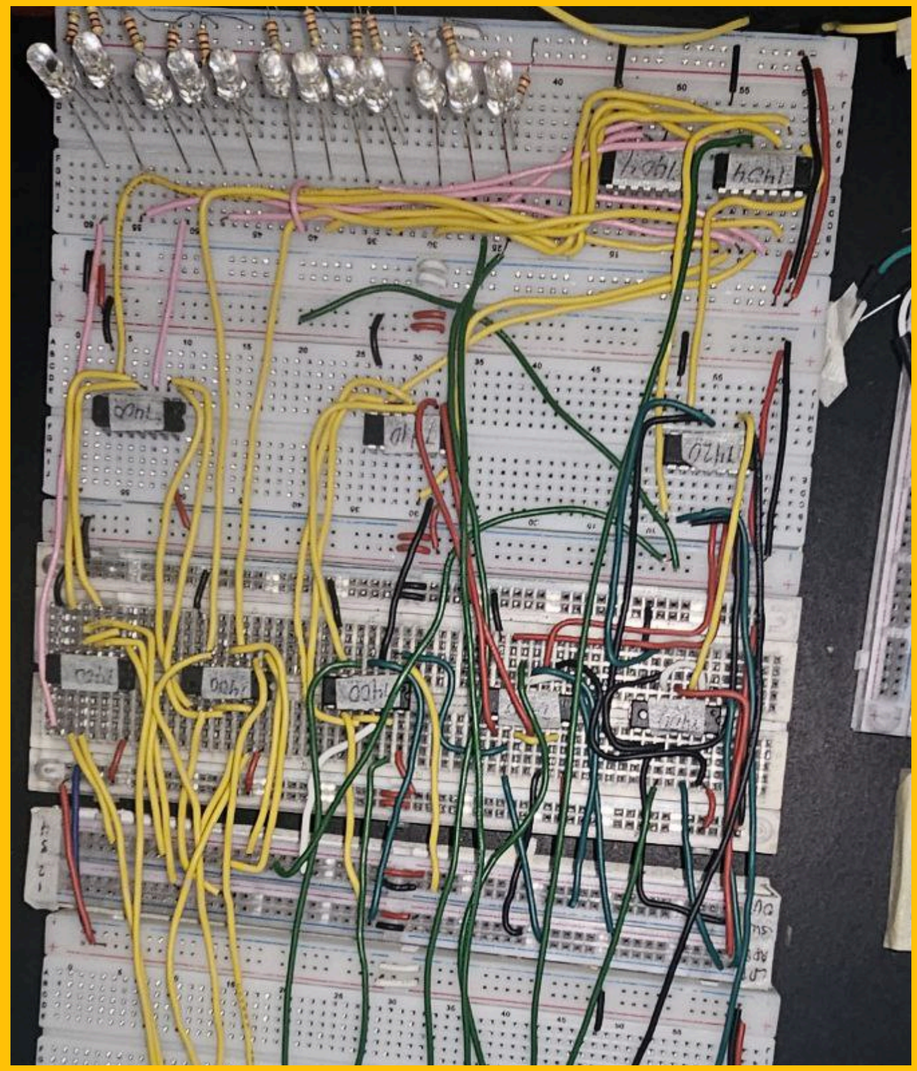
- A circuit that generates specific control signals based on the opcode and ring counter.

Why they’re needed:

- It activates the right components at the right time.
- Works like a conductor directing an orchestra.

How they work:

1. Receives input from the **Instruction Decoder** and **Ring Counter.**
2. Outputs control signals to registers, ALU, and memory.



Control Matrix Instruction

All Instruction
T1 5E3H
T2 BE3H
T3 263H

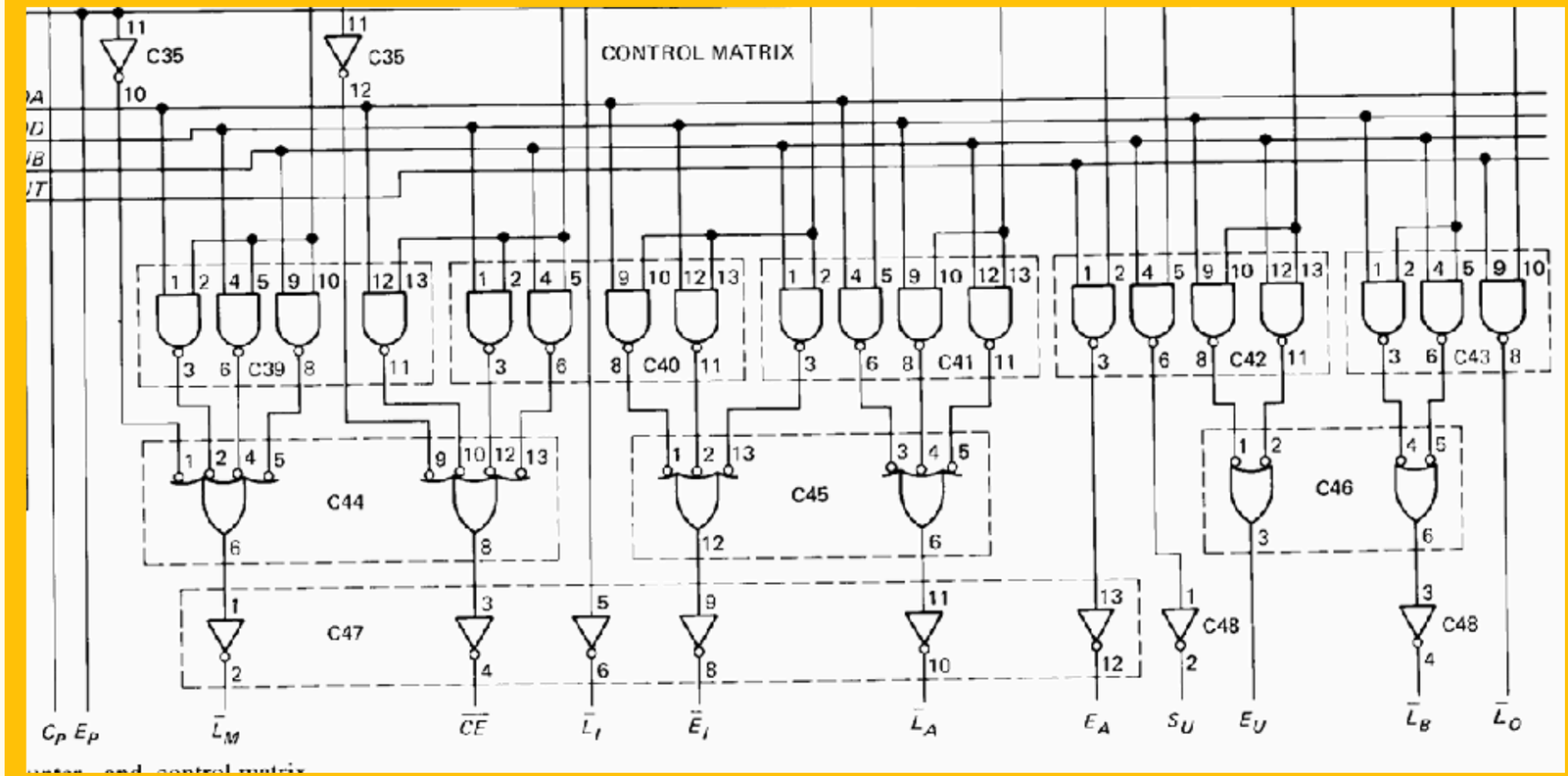
LDA
T4 1A3H
T5 2C3H
T6 3E3H

ADD
T4 1A3H
T5 2E1H
T6 3C7H

SUB
T4 1A3H
T5 2E1H
T6 3CFH

OUT
T4 3F2H
T5 3E3H
T6 3E3H

HLT
T4 3E3H



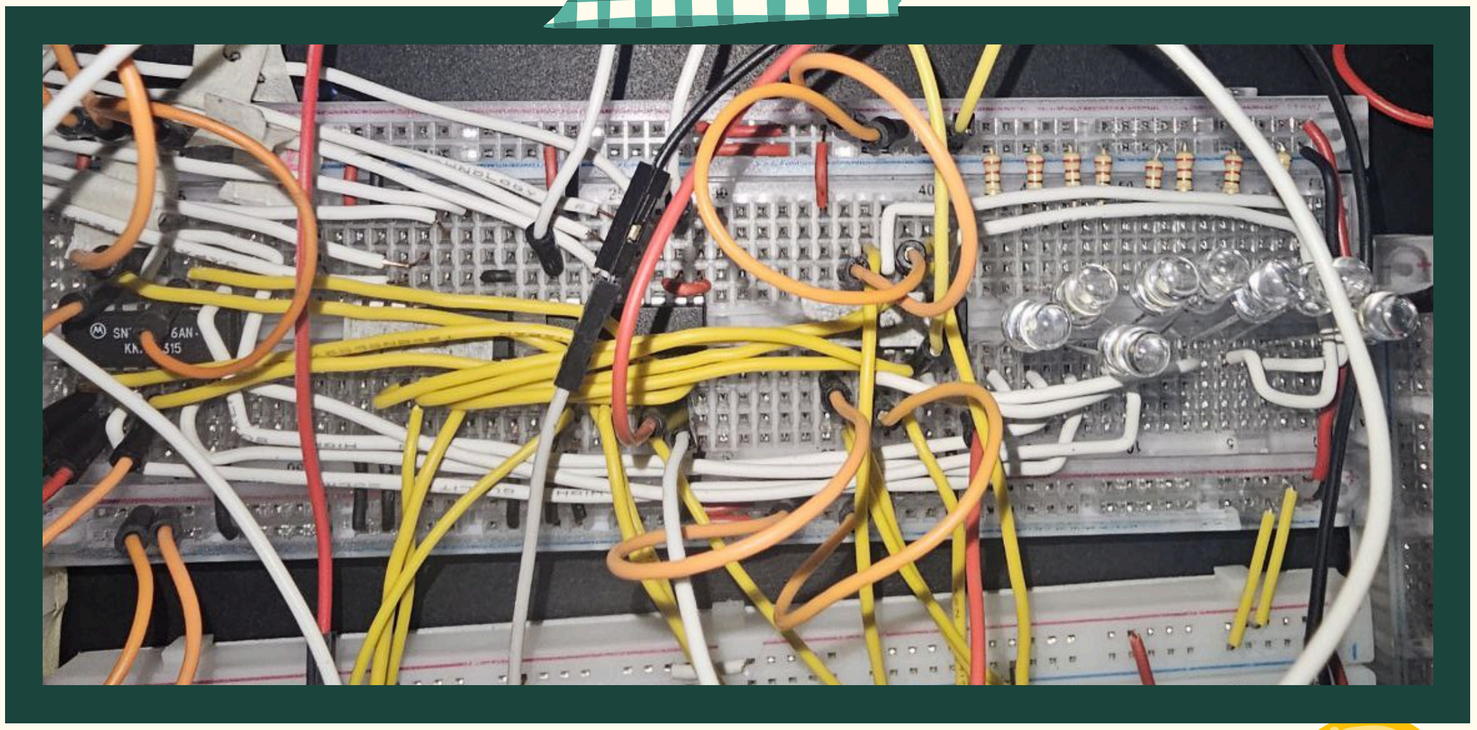
Accumulator (A Register)

What it is:

- A special register that holds the results of arithmetic and logical operations

Why it's needed:

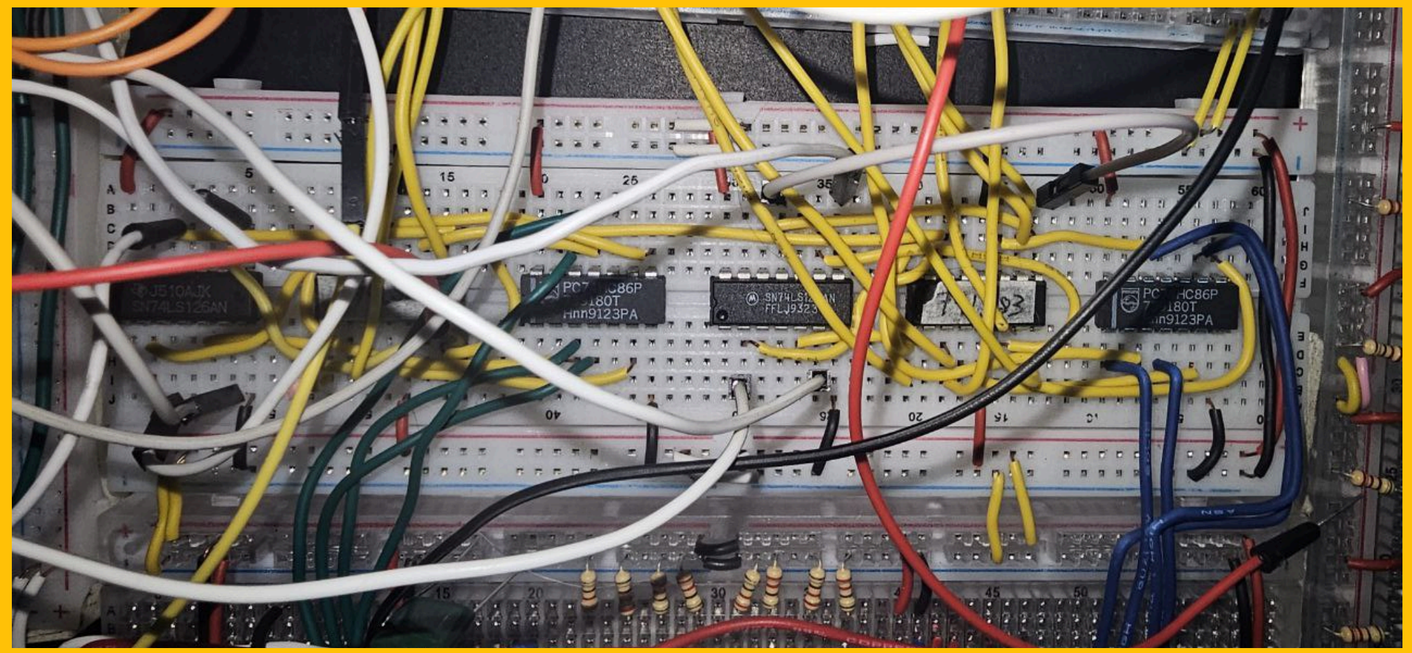
- Acts as **temporary storage** for ongoing calculations.
- Essential for performing multi-step computations.



How it works:

1. The ALU performs an operation.
2. The result is stored in the Accumulator for further use.

Adder/Subtractor (ALU - Arithmetic Logic Unit)



What it is:

- The brain of arithmetic operations.
- Performs addition and subtraction.

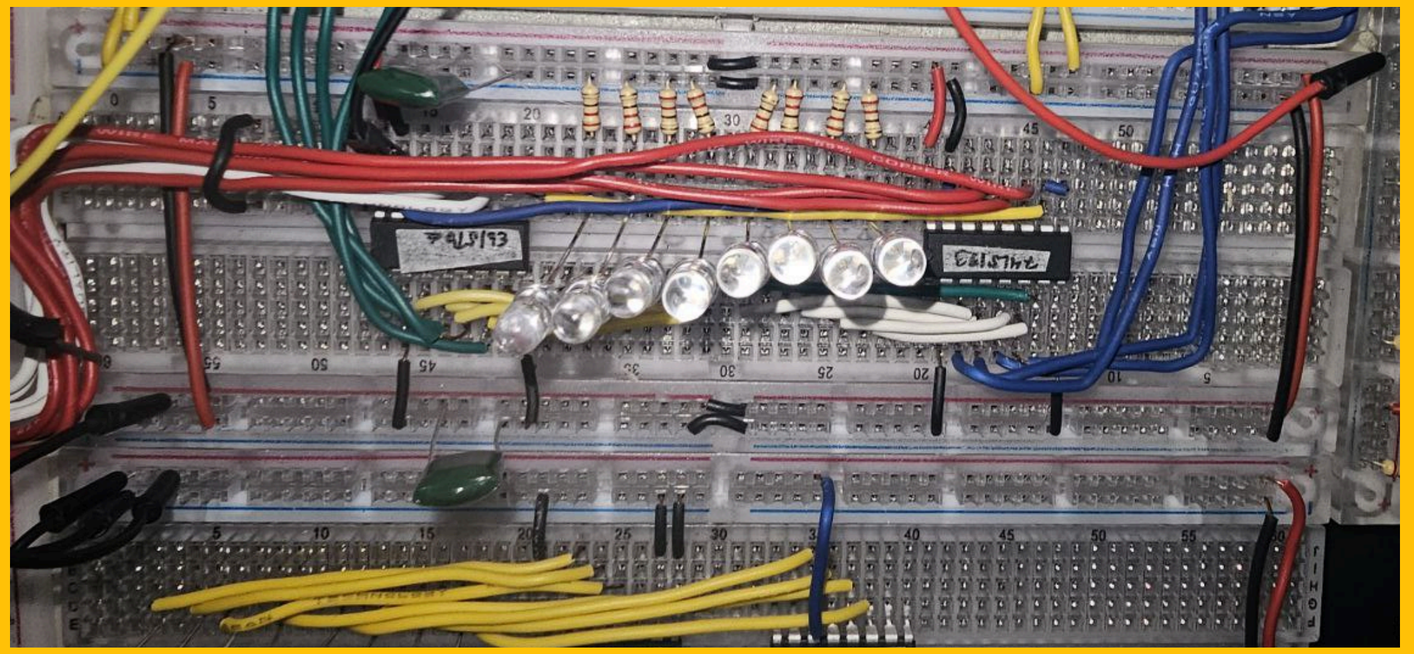
Why it’s needed:

- Essential for executing mathematical calculations.

How it works:

1. Takes inputs from the Accumulator (A Register) and B Register.
2. Adds or subtracts based on the instruction.
3. Stores the result back into the Accumulator.

B Register



What it is:

- A register that **holds the second operand** for ALU operations.

Why it’s needed:

- ALU operations require two values, and the B register supplies one of them.

How it works:

1. Loads data from memory or W Bus.
2. Passes it to the ALU for computation.

Output Register

What they are:

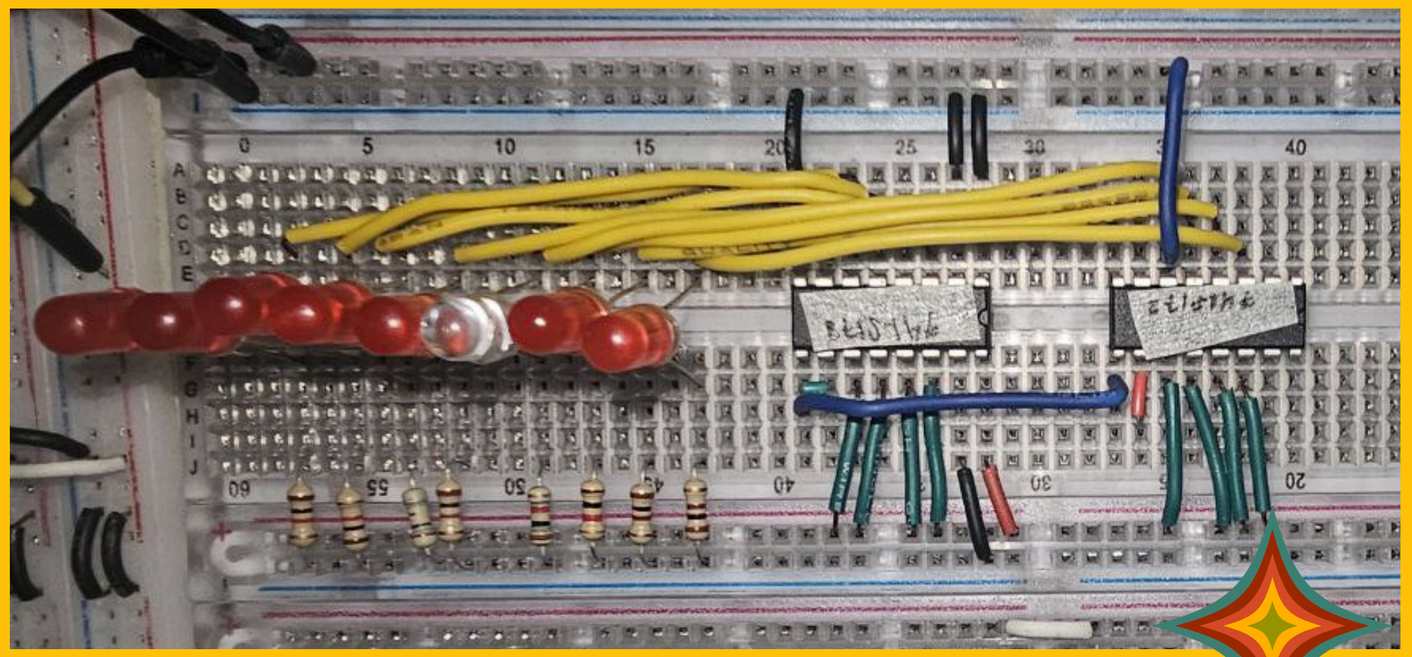
- Stores the **final computed result** before sending it to an external display.

Why they’re needed:

- Ensures that results are properly held before being shown.

How they work:

1. The final value from the Accumulator is stored here.
2. Is then displayed as output.



Thank you!
for listening!

SIMPLE AS POSSIBLE COMPUTER -1