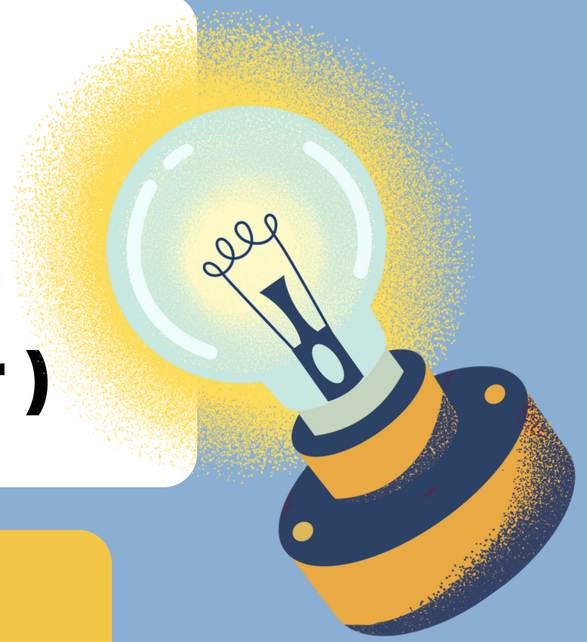
SAP-1

(Simple As Possible-1 Computer)



Presentend by: **ByteSquad**

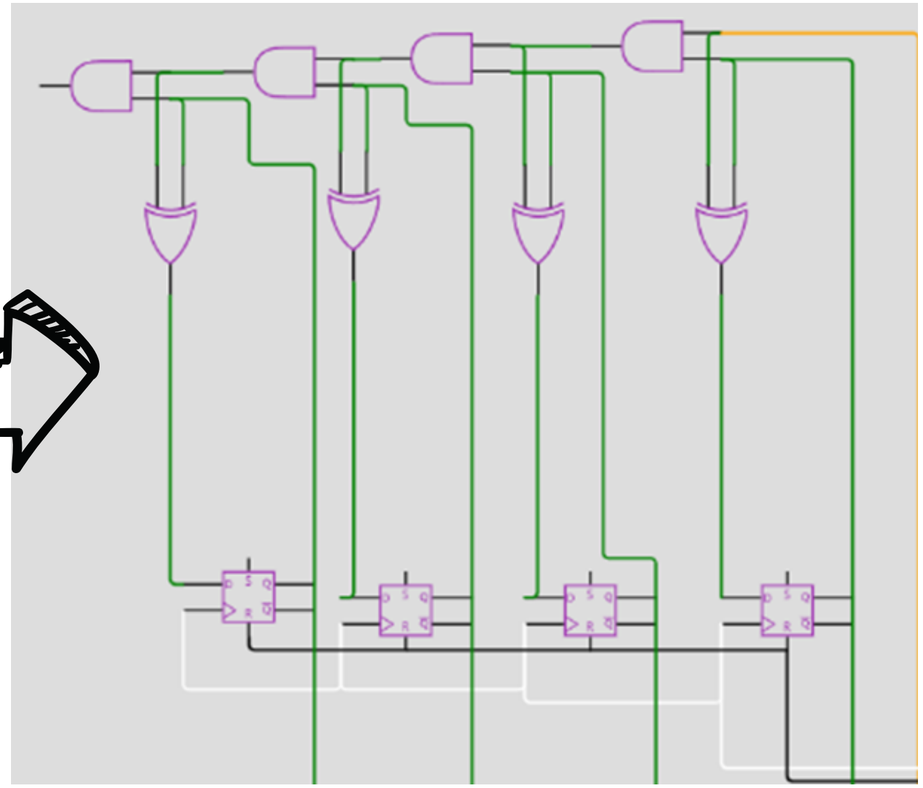
Objectives

- Learn to implement a Simple-As-Possible (SAP 1) computer with all its components
- To understand the functionality of how the components of SAP-1, which are IR, ALU, Accumulator, Program Counter, etc come together and operate a computer.
- To understand how these components are designed on Wokwi and their extent of working.

Which components is our SAP-1 using?

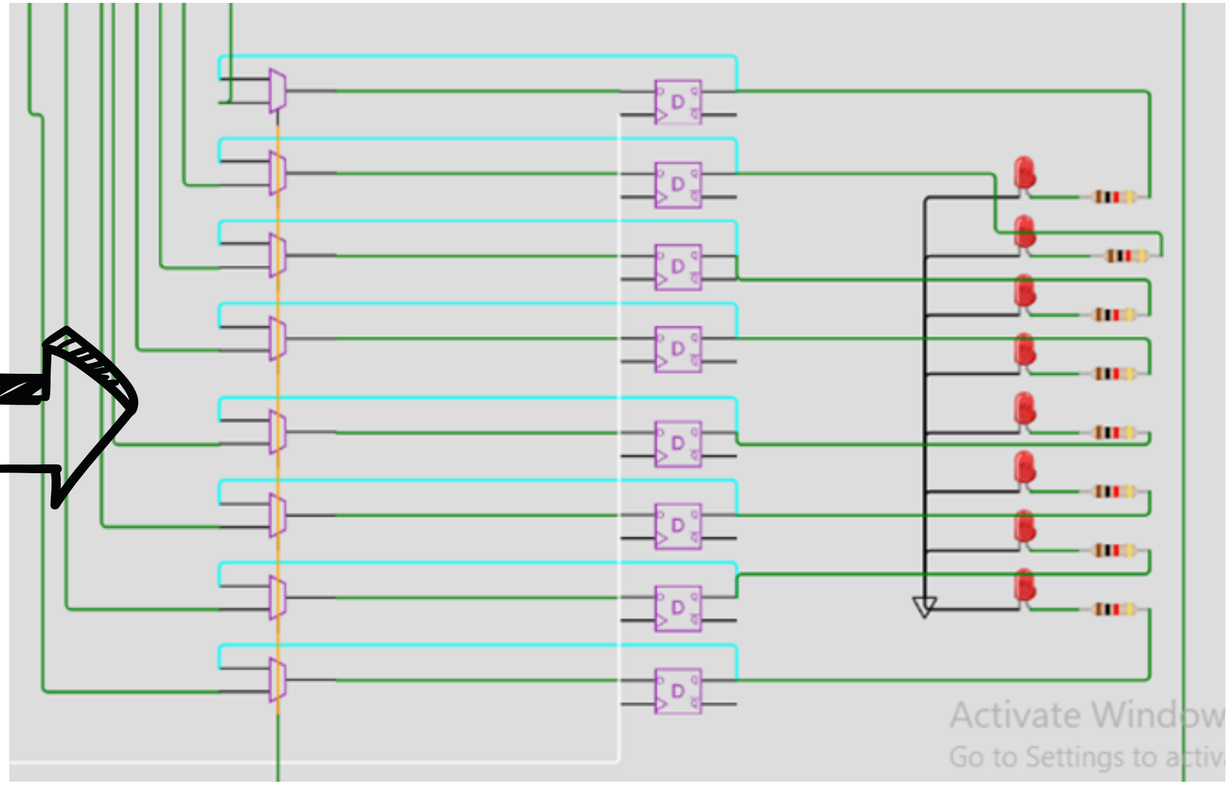
4-bit Program Counter

- Comprises D-flip flops (store current value of the program counter).
- This value is updated on the basis of control signals such as increment, load and reset.
- Clock source provides clock pulses for synchronization & step-by-step execution.
- clock pulse updates the program counter at every positive clock edge.
- Also includes Half Adders that perform the incrementing mechanism after every instruction. clock cycle.
- The inputinclude a Clock signal (CLK) and Clear signal (CLR) - resets the PC to 0.
- The outputs of a program counter include a 4-bit memory address, that is being read from or written to.
- The output is connected to the W bus, however to test and visualize the current value of the component itself we used LEDs.



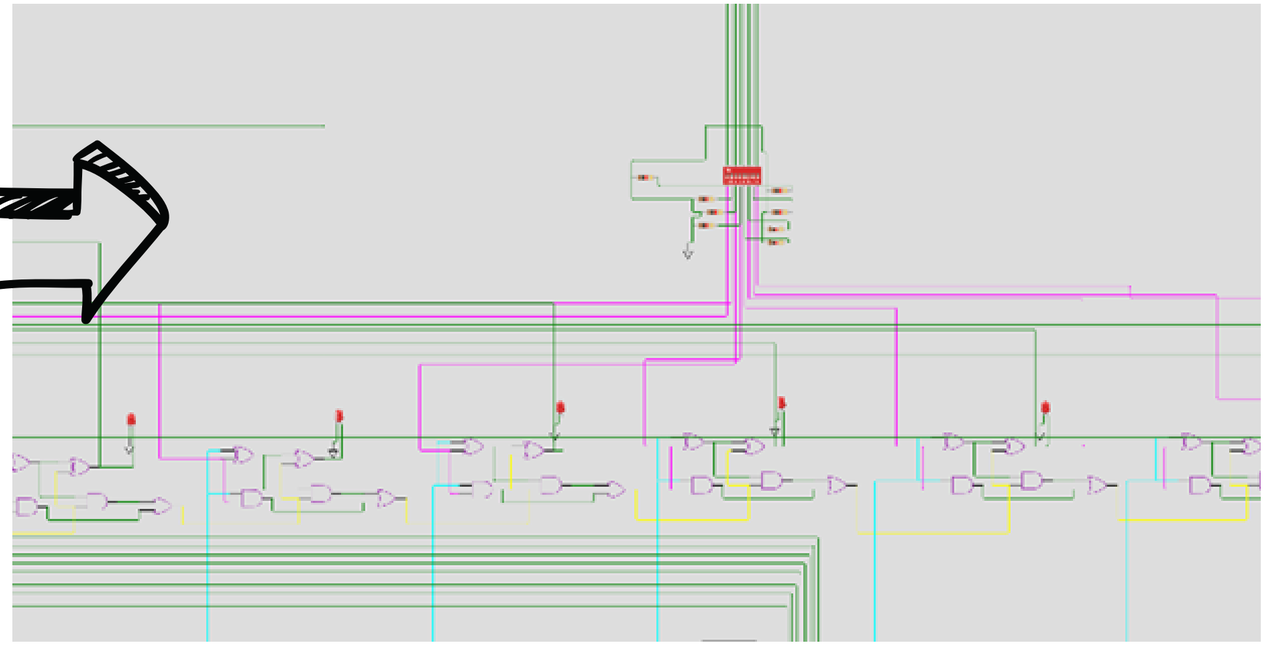
8-bit Instruction Register

- Receives 8-bit instructions via the W bus (data bus).
- pathway through which external instructions are loaded into the registers.
- It requires control signals such as the clock (CLK) for updating the register contents and the clear (CLR) signal for resetting the register contents.
- operates in synchronization with the clock signal, ensuring precise timing for the retrieval and decoding of instructions.
- A 4 bit output is given to the W bus (data bus)
- The sequencer scheduler (micro-controller) takes the 4 bit opcode from the Instruction Register and generates control signal based on this opcode.
- These specifications ensure that the instruction register effectively stores, splits and transfers the instructions within the Control Unit contributing to the overall functionality of the system.
- Jump command, when applied, moves to the next command in Program Counter and loads it onto the Instruction Register.



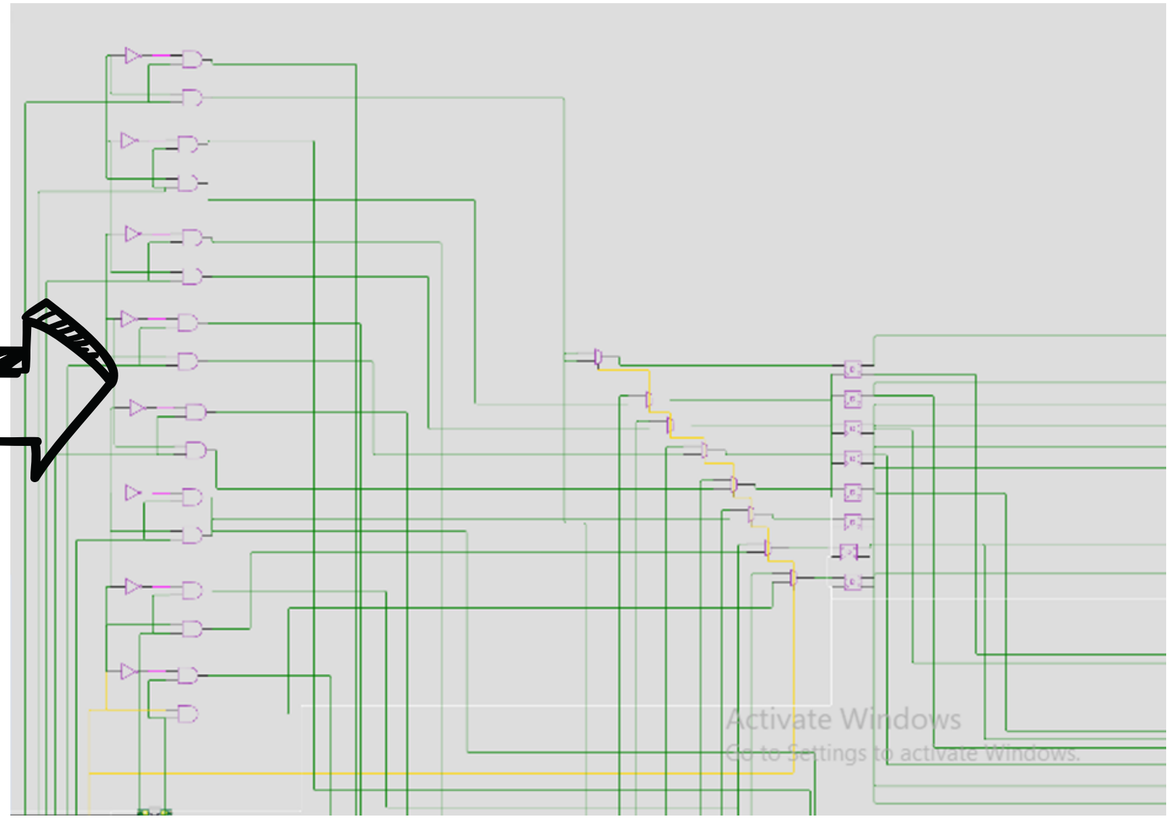
Arithmetic Logic Unit (8-bit Adder)

- The ALU can function as an 8 bit adder/subtractor.
- We're implementing an ALU that only performs addition.
- It uses 8 XOR gates and 8 Full Adders to function as an 8 bit adder.
- Receives two values from the accumulator and the demux, performs addition on them and sends the result out to the W bus.
- Since, we're not using flags in our SAP-1 model, the values to be added must not add to result in a value with an overflow bit.
- The Adder takes inputs form Accumulator and the demultiplexers that hold the second value to be added.
- The board has one of the switches connected to the Ground in order to perform addition.
- For subtraction to be possible, for example, we had to make that switch high and implement accordingly.
- The result after the operation is stored in the Accumulator by mux.
- When mux select line is high, result gets loaded in accumulator.



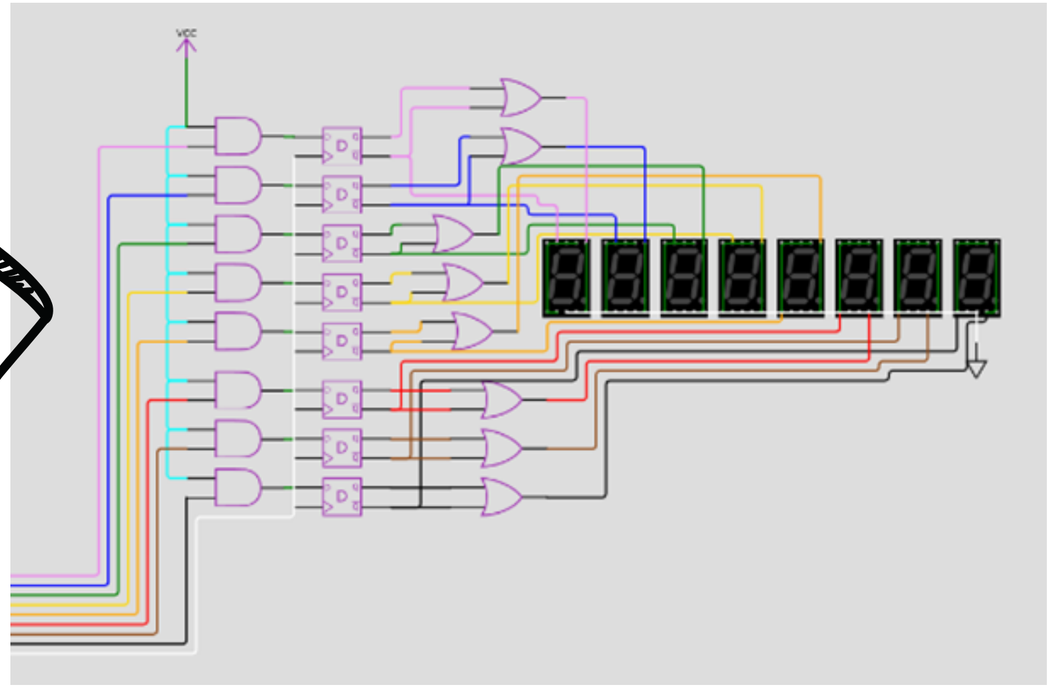
Accumulator

- 8-bit Accumulator incorporates 8 D-Flip Flops, 8 2:1 MUX, and 8 2:1 DEMUX components.
- The DEMUX directs data flow, determining whether to load data into the Accumulator or the B register based on the microcontroller's instruction set.
- When DEMUX is set to 0, data loads into the Accumulator via the 2:1 MUX.
- When DEMUX is set to 1, the data loads into the second input of the ALU, effectively acting as the B-Register.
- The 8 2:1 MUX determines the data destination for the Accumulator;
- when MUX set to 0, it loads the addition result into the 8 D Flip-Flops of ACC
- when MUX is set to 1, it loads data from memory through DEMUX into ACC.
- The 8-bit output from the Accumulator simultaneously feeds into both the ALU and the Output Register.



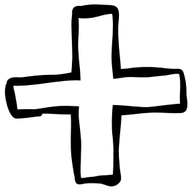
Output Register + 7-seg Binary Display

- Output register stores the output value stored in accumulator as a result of an arithmetic operation done by ALU.
- The bus loads this output value to the output register.
- Binary Display on the top block is a row of eight LEDs.
- It shows content of output by connecting each LED to outputs on the output register. It allows us to check the output taken from accumulator in binary.



RAM + Micro- Controller

We have implemented a 16x8 Random Access Memory through a 2d array.
The first half of the RAM contains the program that the computer runs, while the later half contains the data to be used by the program.
The program counter counts from 1 to 7 currently, since we 8- 15 address in the memory have been occupied by the data .



Instruction Set:

LDA (Load data in Acc)
ADD (Add data with Acc data)
JUMP (Jump to this address)
HLT (Halt Program, PC resets)
OUT (Load value in Acc to Out)
SUB (Sub data with Acc data)

Modifications made to our SAP !

B-Register has been implemented through DMUX and not as a separate data register.

When the Select line signal is 1, data from the micro-controller is loaded into the Accumulator through the MUX.
However, when the Select line signal is 0, the DMUX loads the second data (Data B) from the RAM, providing it as the second input to the ALU.

Upon testing our SAP we realized that the function of the MAR could also be implemented through the PC.
This improved our overall architecture, the latency encountered while using Wokwi and it will reduce the cost if this SAP was implemented in a real world scenario!

What we couldn’t do :(

The SUB instruction in our SAP has been implemented through the code, however we started on with making our ALU solely for the purpose of adding. Hence this extended functionality could not be perfectly simulated.

The Output Register in our SAP requires an enable signal, however unfortunately we had used up all the General Purpose (GP) pins of our micro controller board.
At the moment it is given VCC, so it always displays the value in the accumulator correctly, if you pause and look at the 7 seg binary display connected to it.

Thankyou for Listening!

We’ll now proceed with running our SAP-1 for you!