Computer Architecture Project 2023

Group 14

Moxesh Shah - 2020UEC1080

Prakhar Agrawal - 2020UEC1135

Vatsalya Sagraya - 2020UEC1198

Khushbu Swami - 2020UEC1706

Hitakshi Baharwani - 2020UEC1710

link for the simulation of the program

Introduction to the Architecture of our Processor

- Our Processor includes the following elements with proper sizing:

RAM Size = 64 * 8

bit Register Used: A, B

Temp Register Size: 8 bits

Internal Data Bus: 8 bits

Instruction Format:

I	OPCODE	ADDRESS

- The functional Units of our processor include:

1. Accumulator
2. Program counter
3. Register B
4. ALU
5. Temporary register
6. ROM
7. RAM
8. Instruction Register
9. Memory access register

- Basic processor architecture is used with some modifications.

  Registers used:

  1. Accumulator: 8-bit register. The result is stored in the accumulator.
  2. PC: PC stands for program counter. PC is 8-bit. It stores the location of the next instruction to be executed.
  3. TEMP: TEMP is 8 bits. TEMP is used to store a value temporarily.
  4. Register B: B is 8 bits.
  5. MAR: MAR is 8-bit.
  6. IR: Instruction register is 8-bit.

- React-based design: Our project is built using ReactJS, a popular and widely-used JavaScript library for building user interfaces. This design choice allows for a more modular and reusable codebase, as well as a more streamlined development process.
- JavaScript-based Implementation: By using JavaScript to build out the processor's functionality, we were able to create a more modular and scalable architecture that is easy to modify and extend over time.

Unique Features of NovaTech:

- Dynamic visualizations: Our project provides users with dynamic and interactive visualization of the processor's operation. By displaying each step of the process in real time, users can gain a deeper understanding of how the processor works and how their code is executed.
- Register swapping: One of the key features of our processor is the ability to swap the values of two registers. This feature can be useful for a variety of applications, from sorting algorithms to optimizing code execution.
- Clock cycle tracking: Our project also includes a clock cycle tracker that displays the current clock cycle and updates in real time as the processor executes code. This feature provides users with a clear understanding of how long each step of the process takes and can be helpful for debugging and optimization.

Instruction Set

SR NO:	MNEMONICS	DESCRIPTION	FLAGS AFFECTED	OPCODE
1	MOV A, B	MOVE CONTENTS FROM ONE REGISTER TO ANOTHER	NONE	0X00
2	MVI A	MOVE IMMEDIATE TO A	NONE	0X01
3	LDA X	LOAD CONTENTS TO A FROM MEMORY	NONE	0X30
4	ADD A, B	ADD B TO A	NONE	0X02
5	SUB	SUBTRACT B FROM A	NONE	0X96
6	AND	AND THE VALUES OF BOTH REGISTERS	NONE	0X03
7	OR	OR THE VALUES OF BOTH THE REGISTERS	NONE	0X25
8	INR A	INCREMENT A	NONE	0X04
9	DCR A	DECREMENT A	NONE	0X05
10	CMA	COMPLIMENT OF A	NONE	0X08
11	SWAP A, B	SWAP VALUES OF A AND B	NONE	0X09
12	SQR A	SQUARE OF A REGISTER	NONE	0X0B
13	HCF A, B	SQUARE ROOT OF A REGISTER	NONE	0X0C
14	LCM A, B	LOWEST COMMON MULTIPLE OF A AND B	NONE	0X0D

Micro-instruction for Each instruction set

1. MVI:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: RAM(E), A(L)
T3:
T4:
T5:
T6:

2. MOV:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: A(E), B(L)
T3:
T4:
T5:
T6:

3. ADD:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: B(E), A (E, L), A=A+B
T3:
T4:
T5:
T6:

4. AND:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: B(E), A(E, L), A=A&B
T3:
T4:
T5:
T6:

5. SUB:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: B(E), A(E, L), A=A-B
T3:
T4:
T5:
T6:

6. OR:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: B(E), A(E, L), A=A||B
T3:
T4:
T5:
T6:

7. INR:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: A (E, L), A=A+1
T3:
T4:
T5:
T6:

8. DCR:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: A (E, L), A=A-1
T3:
T4:
T5:
T6:

9. SQR:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: A(E, L), A=A*A
T3:
T4:
T5:
T6:

10. SWAP:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2:
T3:
T4: B(E), TMP(L)
T5: A(E), B(L)
T6: A(E), B(L)

11. LCM:

T0: PC(E), MAR (E, L)
T1: PC(I), RAM(E), IR(L)
T2: B(E), A(E, L), LCM(A,B)
T3:
T4:
T5:
T6:

12. HCF:

T0: PC(E), MAR (E, L)

T1: PC(I), RAM(E), IR(L)

T2: B(E), A(E, L), HCF(A,B)

T3:

T4:

T5:

T6:

13. CMA:

T0: PC(E), MAR (E, L)

T1: PC(I), RAM(E), IR(L)

T2: A (E, L), A=A’

T3:

T4:

T5:

T6:

14. LDA:

T0: PC(E), MAR (E, L)

T1: PC(I), RAM(E), IR(L)

T2: PC(E), MAR (E, L)

T3: PC(I), RAM(E), TEMP(L)

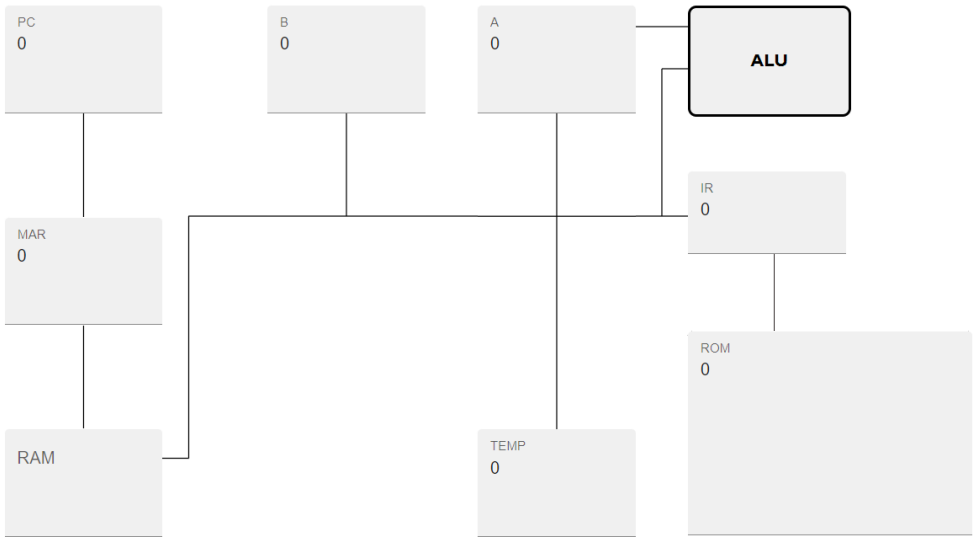
T4: TEMP(E), A(L)

T5:

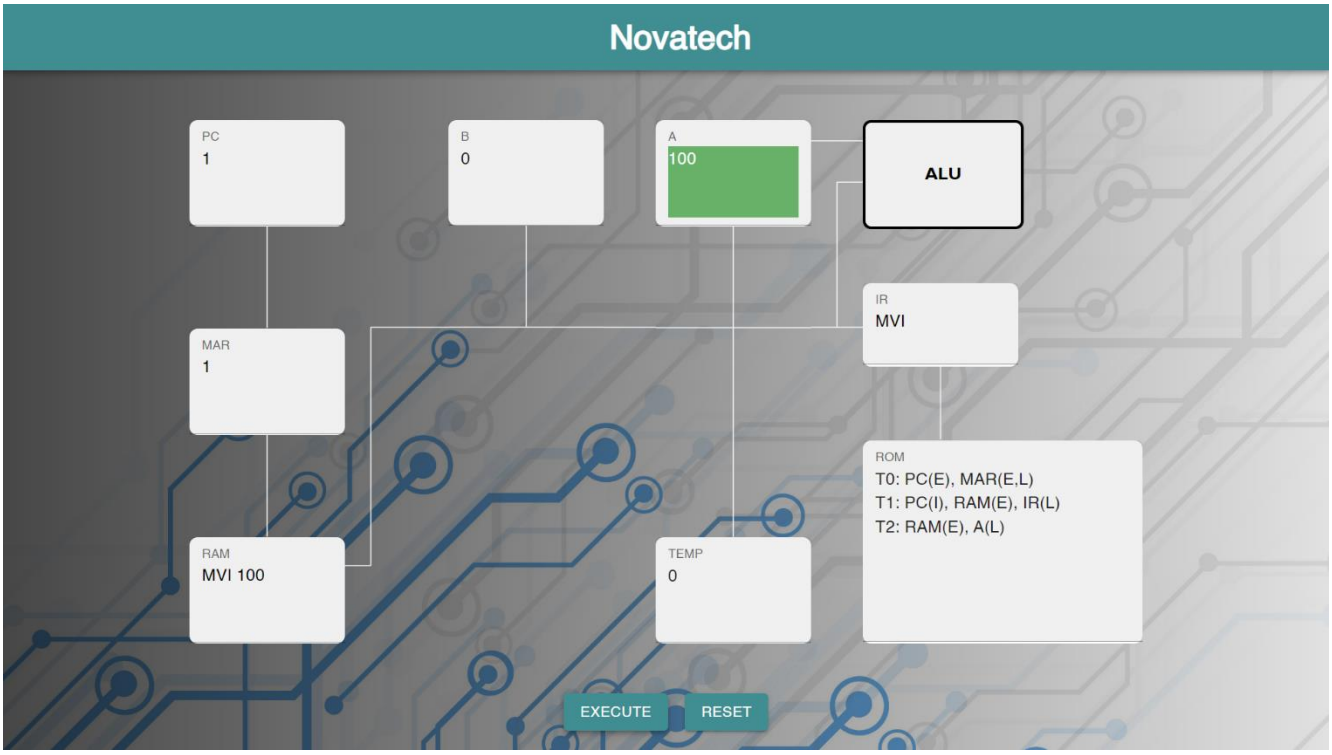
T6:

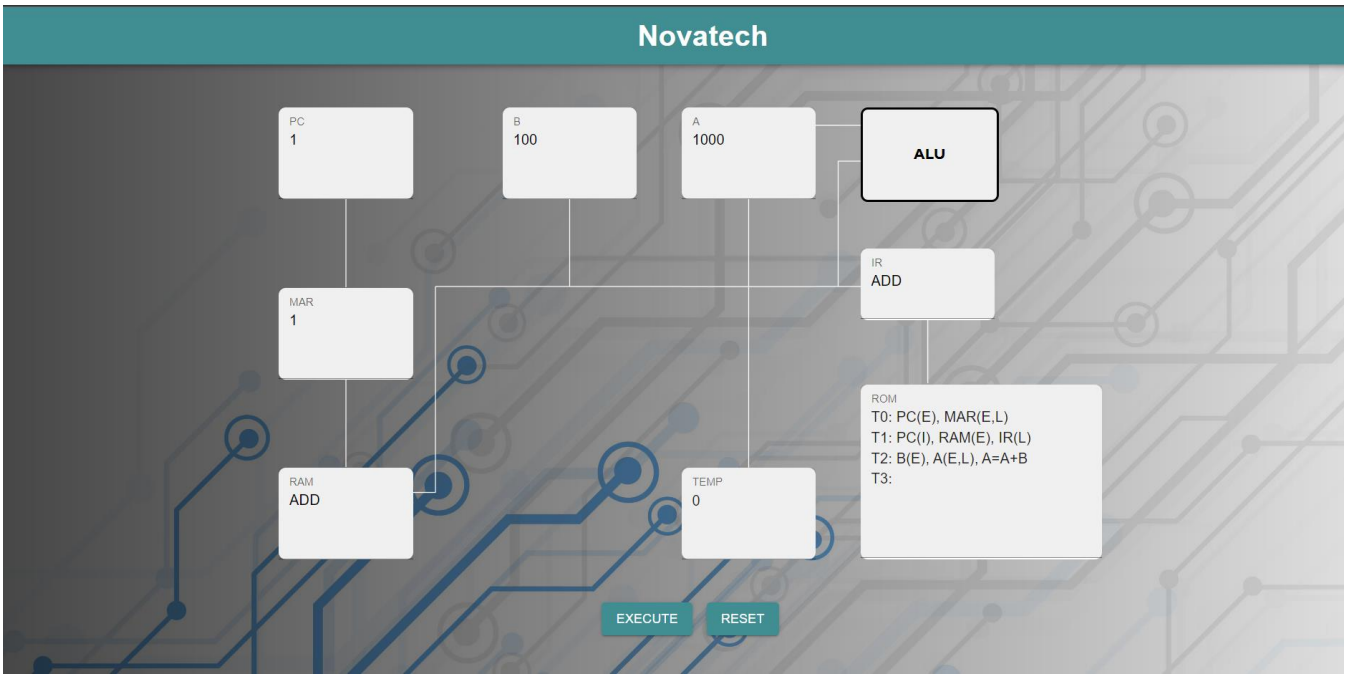
Working with Diagram

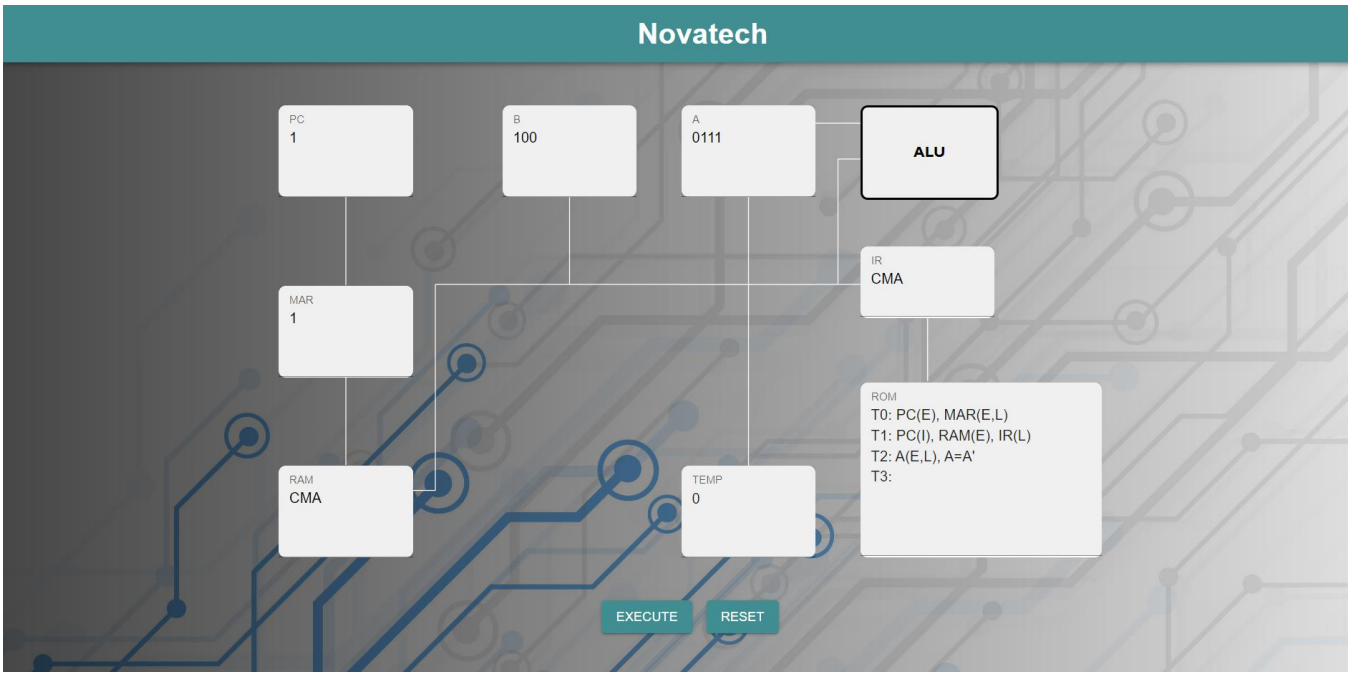
Architecture of our Processor:



Screenshots of Output of the Simulation of the Program







- **<u>Educational value:</u>** Overall, our project has significant educational value for anyone looking to learn more about processor architecture and operation. By providing a clear and interactive display of the processor's workings, users can gain a deeper understanding of this complex topic.