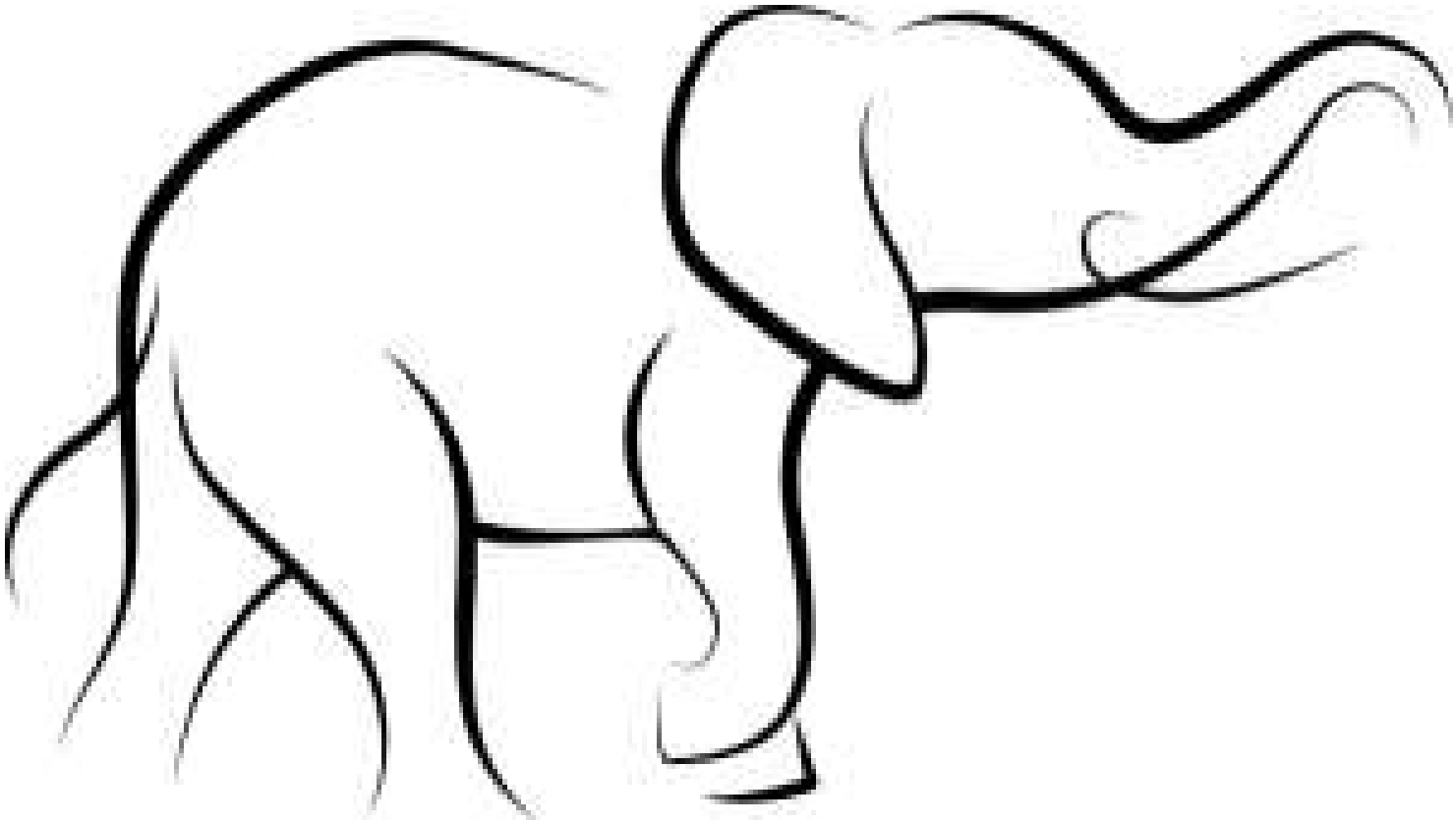
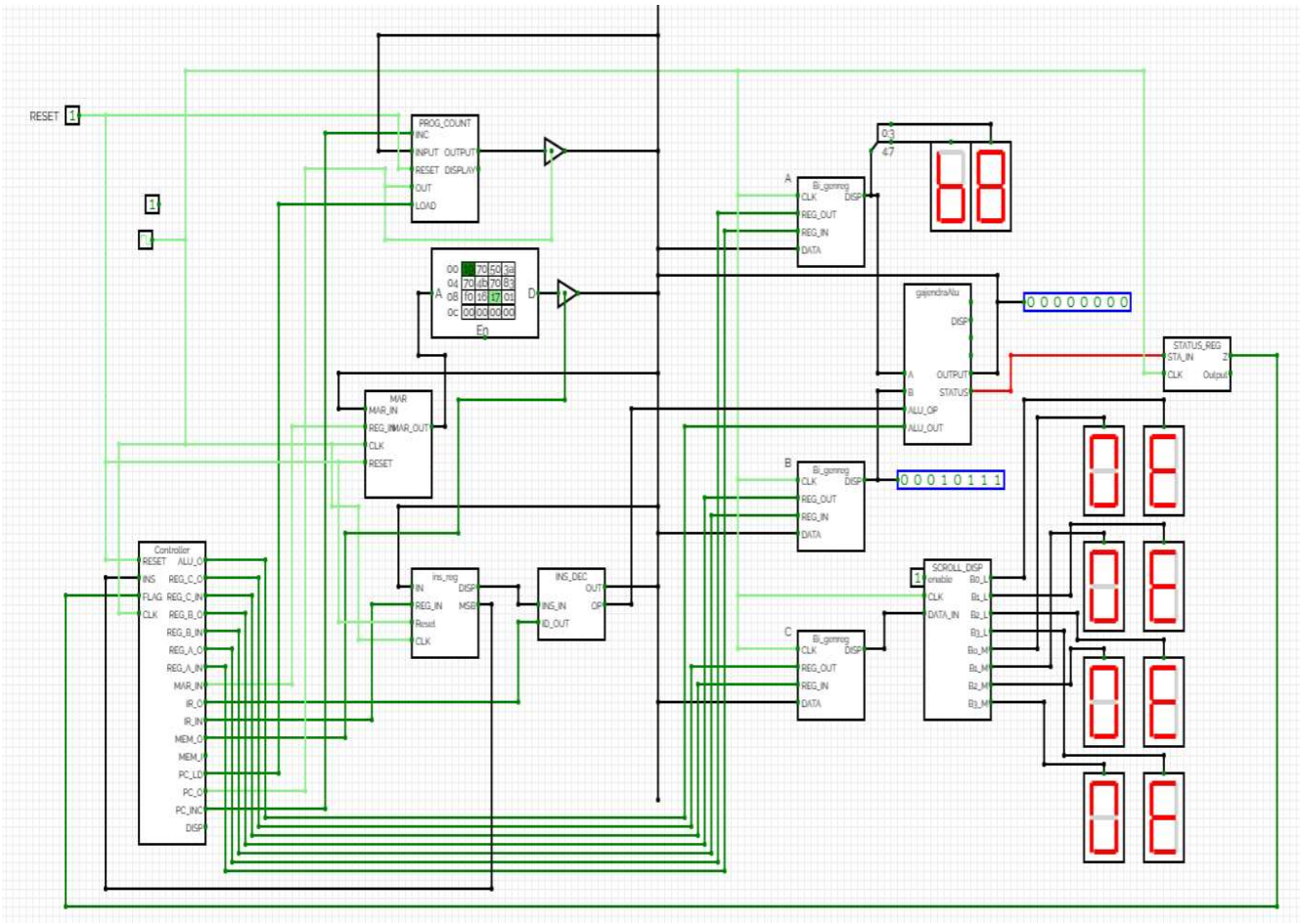
GAJENDRA



By- Pramodh

Sarath

Under guidance of - Ayon sir



FINAL CIRCUIT => GAJENDRA-1

Contents

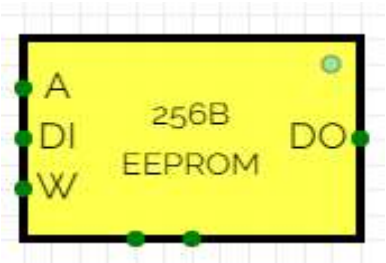
- JNZ - Jump Non-Zero
- MOV AC - Copy data from A to C
- MOV BA - Copy data from B to A
- MOV CB - Copy data from C to B
- MOV AB - Copy data from A to B
- MOV CA - Copy data from C to A
- MOV BC - Copy data from B to C
- HALT - End of instruction

❖ **Assembly Programs implemented using the Instruction set**

❖ **Micro-instructions and Controller Logic Design**

Memory

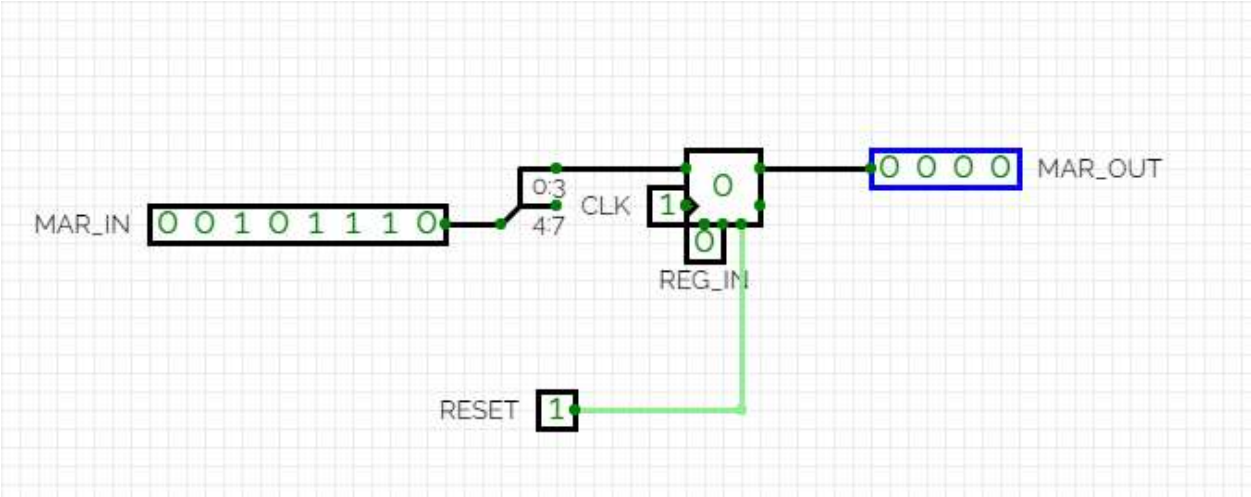
Diagram :



Working : It is an EEPROM. It contains instructions and data.

Memory Address Register (MAR)

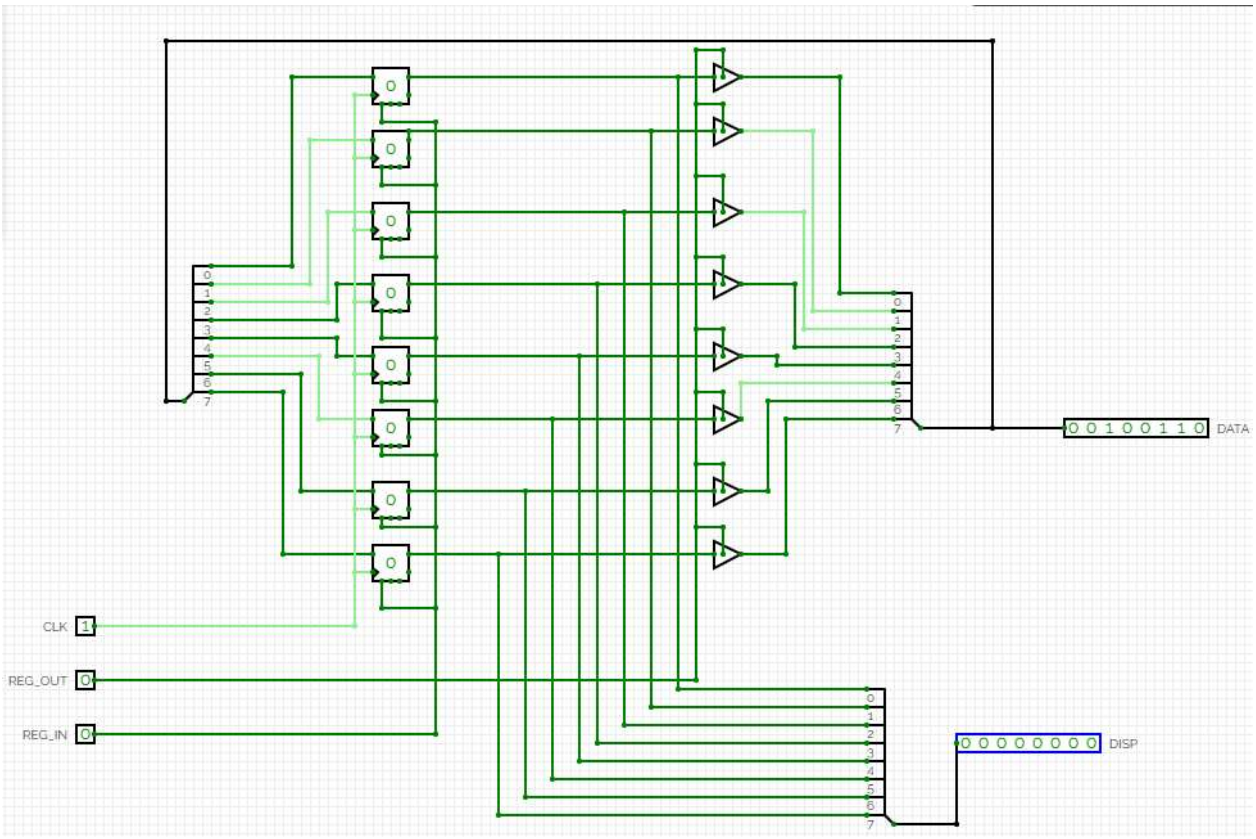
Diagram :



Working : It gives the 4-bit address (LSB of input) of the required data (in memory) by taking 8-bit input for either Program Counter or Instruction Decoder.

General CPU Register

Diagram:

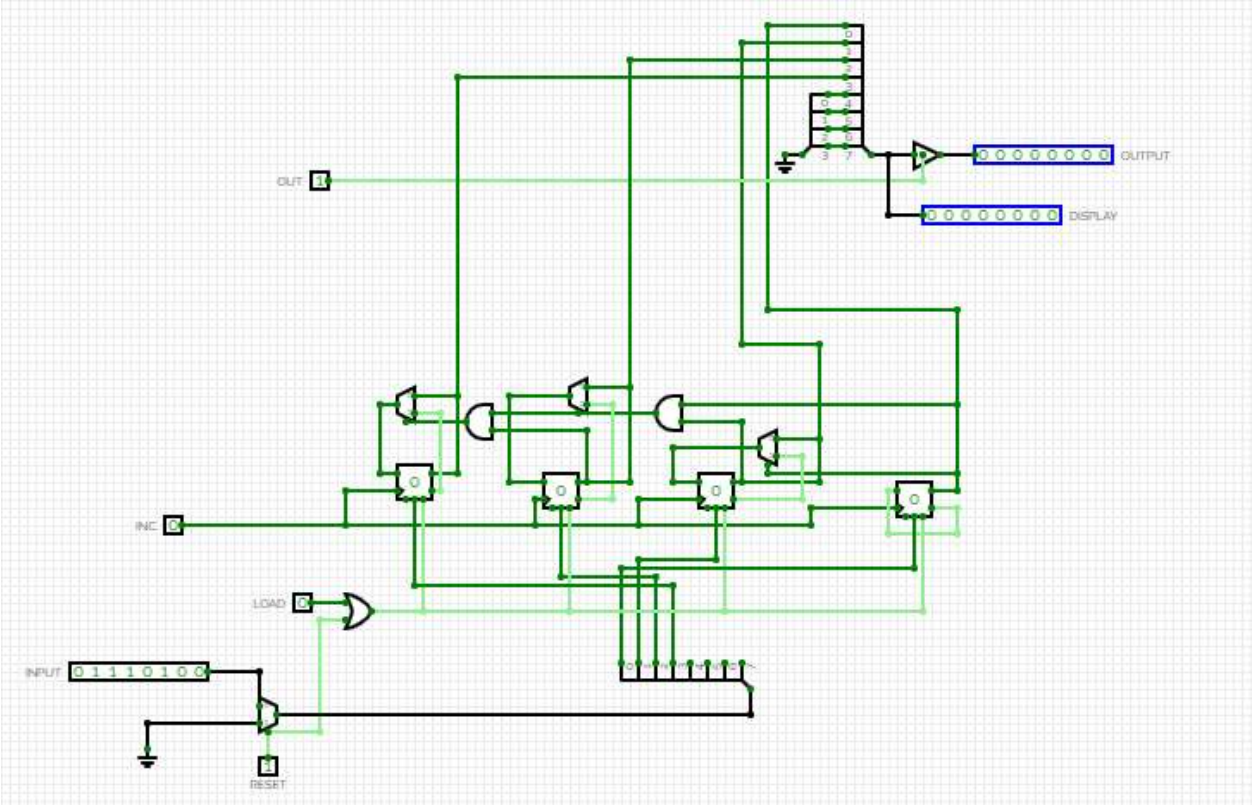


Working:

- When REG_IN is 1(active) , the data is taken as input into the DATA bus.
- When REG_OUT is 1(active) , the data stored in the register will be displayed in the DATA bus.
- DISP will continuously display value whenever an input is given.

Program Counter

Diagram:

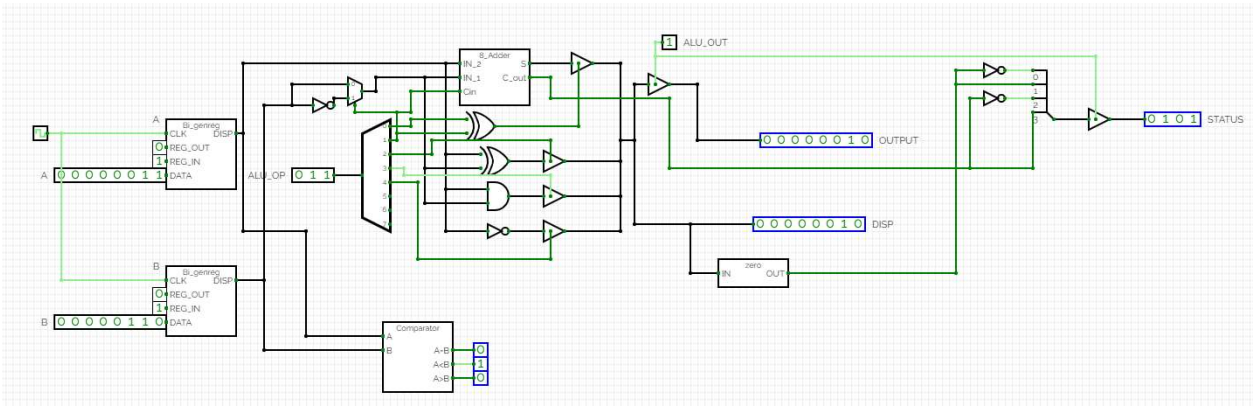


Working:

- The Program Counter points to the address of instruction in ROM.
- When PC_LOAD is set to 1, it loads(takes as input) an 8-bit data from the common bus.
- When PC_Out is set to 1 it gives an output of 8-bits that consists of the address that is to be accessed in Memory.
- When Reset is set to 1, it resets to 0.

Arithmetic and Logical Unit (ALU)

Diagram:



Working:

ALU_OP	OPERATION
000	ADD
001	SUB
010	XOR
011	AND
100	NOT

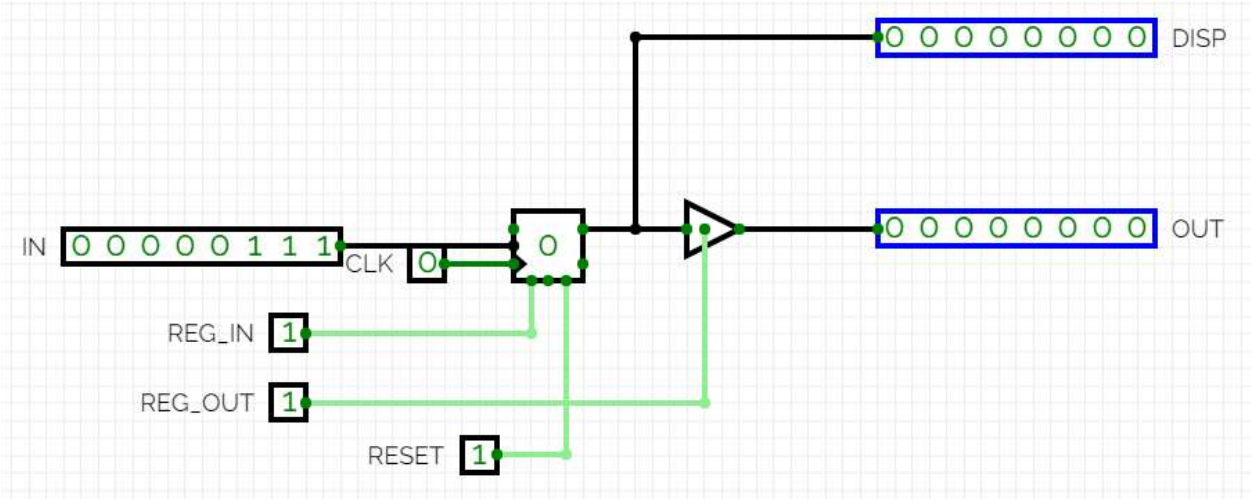
1. A simple ALU supporting 8-bit ADD,SUB,AND,XOR,NOT and COMPARE operations.
2. This ALU can perform the following operations,but we are using only ADD and SUB (Ins Decoder gives inputs to ALU).
3.Here in this ALU, it automatically compares the values A and B every time we pass two values A,B into the ALU, it

displays ‘1’ in the A>B block if A>B, it displays ‘1’ in the A<B block if A<B, else it displays ‘1’ in A=B block if A=B.

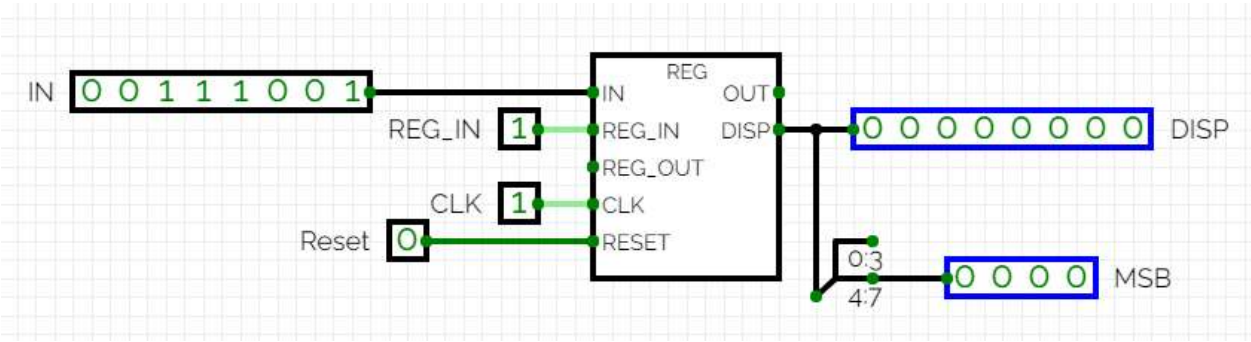
Instruction Register

Diagram:

REG:



INS_REG:



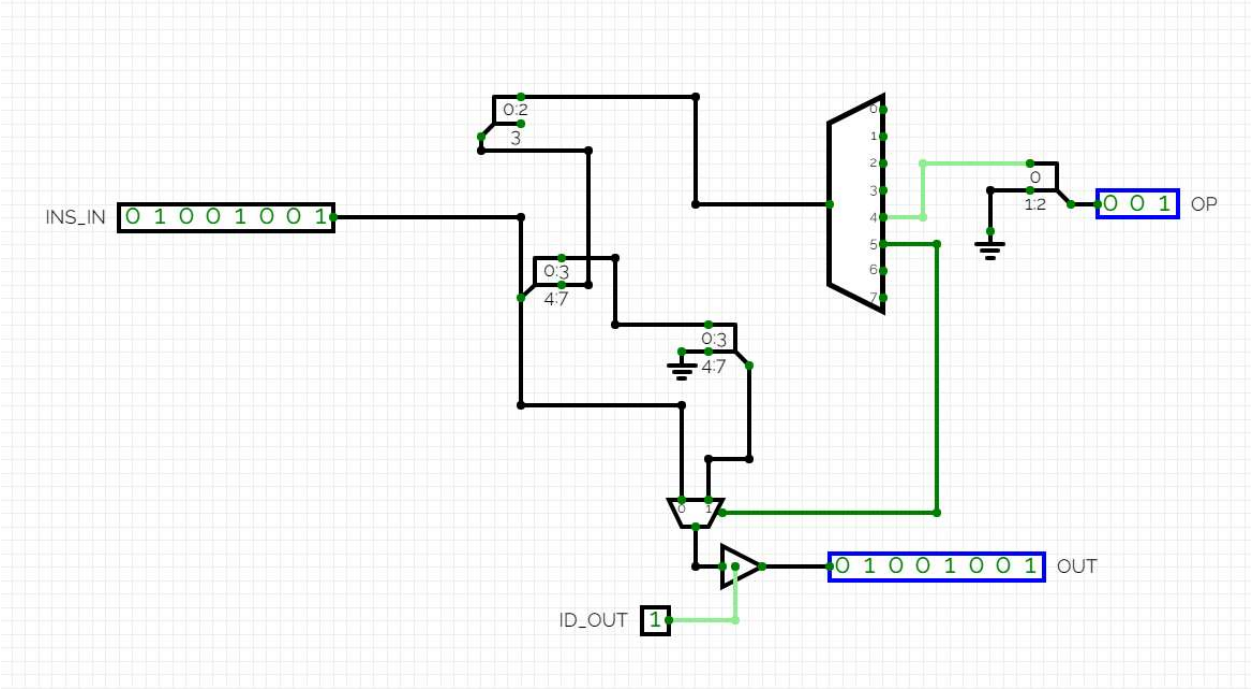
Working:

- It takes input (if IR_IN = 1) from Common Bus (MEM) and gives the same to Instruction Decoder.

- MSB(instruction) of the value stored is given as input for controller.
- If RESET = 1 then it Resets value to 0.

Instruction Decoder

Diagram:



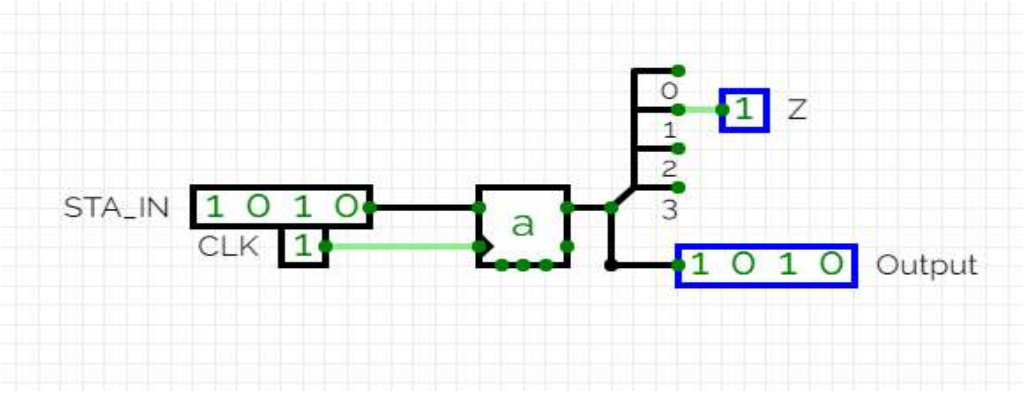
Working:

- It directly takes input from the Instruction Register.
- If the operation is LDI then it changes the MSB(Instruction) of input to 0 without changing LSB(value) else it directly outputs the value that is taken as input.

- If the operation is SUB ,then ALU_OP is given as 1, else 0.

Status Register

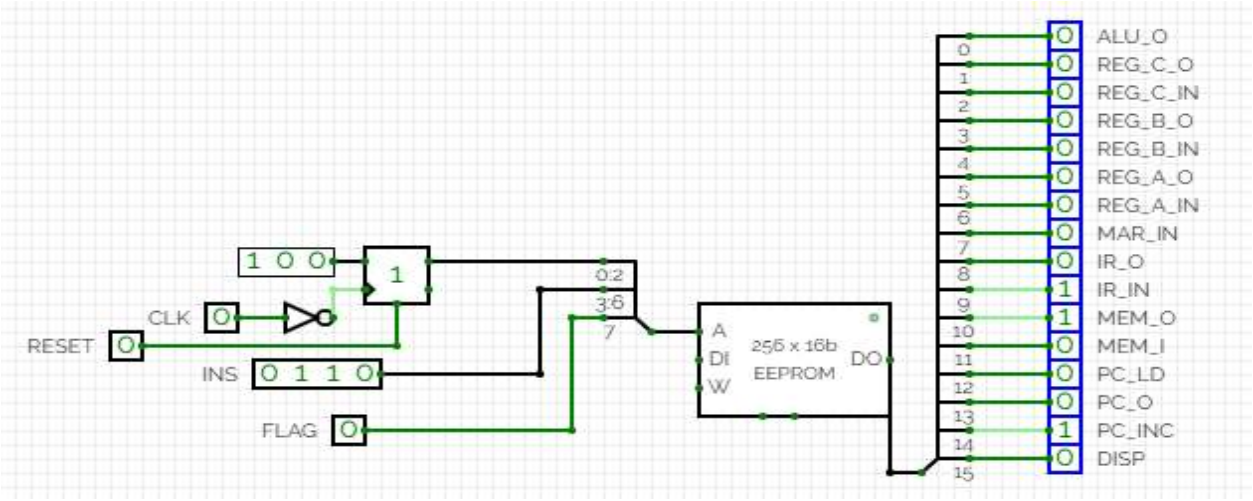
Diagram:



Working:

It takes input from ALU and outputs value of Z that is to be given as flag to controller.

Controller:



It gives controls for every element by taking instruction(from IR) and flag(from Status REG) as input. Counter is used to change T-states.

Instruction and Machine Code

INSTRUCTION	MACHINE CODE
NOP	**0000 0x0**
LDA	**0001 0x1**
STA	**0010 0x2**
ADD	**0011 0x3**
SUB	**0100 0x4**
LDI	**0101 0x5**
JMP	**0110 0x6**
SWAP	**0111 0x7**
JNZ	**1000 0x8**

MOVAC	1001 0x9
MOVBA	1010 0xA
MOVCB	1011 0xB
MOVAB	1100 0xC
MOVCA	1101 0xD
MOVBC	1110 0xE
HALT	1111 0xF

Description of Instruction Set

1.NOP - No Operation

Description - No Operation is performed.

Operation - None.

Syntax	Operands	Program Counter
NOP XXXX	X=0	PC <= PC+1

8-bit Op-Code :

0000	XXXX

2.LDA - Load Accumulator

Description - This instruction loads the accumulator with the contents from the memory by looking at the 4 bit address.

Operation - Ra <= ROM

Syntax	Operands	Program Counter
LDA XXXX	X=0 or X=1	PC <= PC+1

8-bit Op-Code :

0001	XXXX

3.STA - Store at Address A

Description - This takes the value at accumulator and stores at address A.

Operation - ROM <= Ra

Syntax	Operands	Program Counter
STA XXXX	X=0 or X = 1	PC <= PC+1

8-bit Op-Code :

0010	XXXX

4.ADD - Add without Carry

Description - This instruction loads the Reg-B with the value from the memory and then adds up the values of Accumulator and Reg-B and stores the final sum in Accumulator.

Operation -
Rb <= ROM

Ra <= Ra + Rb

Syntax	Operands	Program Counter
ADD XXXX	X=0 or X=1	PC <= PC+1

8-bit Op-Code :

0011	XXXX

5.SUB - Subtraction of two +ve numbers

Description - This instruction loads the Reg-B with the value from the memory and subtracts the value in Reg-B from the accumulator and stores the final value in the accumulator.

Operation -
Rb <= ROM
Ra <= Ra - Rb

Syntax	Operands	Program Counter
SUB XXXX	X=0 or X=1	PC <= PC+1

8-bit Op-Code :

0100	XXXX

6.LDI - Load Immediate

Description - Loads the 4-bit input directly into the accumulator.

Operation - Ra <= k

Syntax	Operands	Program Counter
LDI XXXX	X=0	PC <= PC+1

8-bit Op-Code :

0101	XXXX

7. JMP - Jump to address

Description - This Instruction takes the control to the given memory location.

Operation -

Syntax	Operands	Program Counter
JMP XXXX	X=0 or X=1	PC <= Address(XXXX)

8-bit Op-Code :

0110	XXXX

8.SWAP - Swap A and B

Description - Swaps the values stored in Reg-A and Reg-B with the use of Reg-C(temp).

Operation - Rc <= Ra
Ra <= Rb
Rb <= Rc

Syntax	Operands	Program Counter
SWAP XXXX	X=0	PC <= PC+1

8-bit Op-Code :

0111	XXXX

9.JNZ - Jump when Non-Zero

Description - This Instruction points to the given memory location when the value returned is non-zero after performing an operation.

Operation - If NZ = 1 then ROM Address = XXXX

Syntax	Operands	Program Counter
JNZ XXXX	X=0 or X=1	PC <= PC+1

8-bit Op-Code :

1000	XXXX

10. MOVAC - Copy data from A to C

Description - The values present in Reg-A are loaded into the Reg-C

Operation - Rc <= Ra

Syntax	Operands	Program Counter
MOVAC XXXX	X=0	PC <= PC+1

8-bit Op-Code :

1001	XXXX

11. MOVBA - Copy data from B to A

Description - The values present in Reg-B are loaded into the Reg-A

Operation - Ra <= Rb

Syntax	Operands	Program Counter
MOVBA XXXX	X=0	PC <= PC+1

8-bit Op-Code :

1010	XXXX

12. MOVCB - Copy data from C to B

Description - The values present in Reg-C are loaded into the Reg-B

Operation - Rb <= Rc

Syntax	Operands	Program Counter
MOVCB XXXX	X=0	PC <= PC+1

8-bit Op-Code :

1011	XXXX

13. MOVAB - Copy data from A to B

Description - The values present in Reg-A are loaded into the Reg-B

Operation - Rb <= Ra

Syntax	Operands	Program Counter
MOVAB XXXX	X=0	PC <= PC+1

8-bit Op-Code :

1100	XXXX

14. MOVCA - Copy data from C to A

Description - The values present in Reg-C are loaded into the Reg-A

Operation - Ra <= Rc

Syntax	Operands	Program Counter
MOVCA XXXX	X=0	PC <= PC+1

8-bit Op-Code :

1101	XXXX

15. MOVBC - Copy data from B to C

Description - The values present in Reg-B are loaded into the Reg-C

Operation - Rc <= Rb

Syntax	Operands	Program Counter
MOVBC XXXX	X=0	PC <= PC+1

8-bit Op-Code :

1110	XXXX

16.HALT - End the Instruction

Description - Fetch cycle stops

Syntax	Operands	Program Counter
HALT XXXX	X=0	PC <= PC+1

8-bit Op-Code :

1111	XXXX

Assembly Programs implemented using the Instruction Set

Add two numbers and displaying the result:

Address	Assembly Code	Machine Code
0x0	LDA 0x5	0x15
0x1	ADD 0x6	0x36
0x2	MOVAC 0x0	0x90
0x3	HALT 0x0	0xf0
0x4		0x00
0x5		0x34
0x6		0x12

- Value at address 0X5(34) is loaded to the accumulator.
- Value at address 0X6(12) is added to the value of accumulator (to give 46) and stored in it.
- Value in the accumulator is moved to Reg-C for displaying.

Adding and Subtracting four numbers in some combination

Address	Assembly Code	Machine Code
0x0	LDA 0x5	0x15
0x1	SUB 0x6	0x46
0x2	ADD 0x7	0x37
0x3	SUB 0x8	0x48
0x4	HALT 0xf	0xf0
0x5		0x17
0x6		0x08
0x7		0x25
0x8		0x12

- Value at address 0x5(17) is loaded to the accumulator. Value at address 0x6(08) is subtracted from the value of accumulator(to give 0f) and stored in it.
- Value at address 0x7(25) is added to the value of accumulator(to give 34) and stored in it.
- Value at address 0x8(12) is subtracted from the value of accumulator(to give 22) and stored in it.
- Instructions end here.

Multiplication using repeated addition

Address	Assembly Code	Machine Code
0x0	LDA 0x9	0x19
0x1	SWAP 0x0	0x70
0x2	LDI 0x0	0x50
0x3	ADD 0xa	0x30
0x4	SWAP 0x0	0x70
0x5	SUB 0xb	0x4b
0x6	JNZ 0x3	0x70
0x7	HALT 0x0	0x83
0x8		0xf0
0x9		0x16
0xa		0x17
0xb		0x01

Micro-instructions and Controller Logic Design

NOP:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
0	T2
0	T3
0	T4

Micro-Instruction for NOP.

LDA:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << IR_OUT | 1 << MAR_IN	T2
1 << MEM_OUT|1 << REGA_IN	T3
0	T4

Micro-Instruction for LDA.

STA:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << IR_OUT | 1 << MAR_IN	T2
1 << MEM_IN|1 << REGA_OUT	T3
0	T4

Micro-Instruction for STA.

ADD:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << IR_OUT | 1 << MAR_IN	T2
1 << MEM_OUT|1 << REGB_IN	T3
1<< ALU_OUT| 1 << REGA_IN	T4

Micro-Instruction for ADD.

SUB:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << IR_OUT | 1 << MAR_IN	T2
1 << MEM_OUT|1 << REGB_IN	T3
1<< ALU_OUT| 1 << REGA_IN	T4

Micro-Instruction for SUB.

LDI:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << IR_OUT | 1 << REGA_IN	T2
0	T3
0	T4

Micro-Instruction for LDI.

JMP:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << IR_OUT | 1 << PC_LOAD	T2
0	T3
0	T4

Micro-Instruction for JMP.

SWAP:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << REGA_OUT | 1 << REGB_IN	T2
1 << REGC_OUT|1 << REGA_IN	T3
1 << REGB_OUT| 1 << REGC_IN	T4

Micro-Instruction for SWAP.

JNZ:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << IR_OUT | 1 << PC_LOAD	T2
0	T3
0	T4

Micro-Instruction for JNZ if flag(Z) = 0.

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
0	T2
0	T3
0	T4

Micro-Instruction for JNZ if flag(Z) = 1.

MOVAC:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << REGA_OUT | 1 << REGC_IN	T2
0	T3
0	T4

Micro-Instruction for MOVAC.

MOVBA:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << REGB_OUT | 1 << REGA_IN	T2
0	T3
0	T4

Micro-Instruction for MOVBA.

MOVCB:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << REGC_OUT | 1 << REGB_IN	T2
0	T3
0	T4

Micro-Instruction for MOVCB.

MOVAB:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << REGA_OUT | 1 << REGB_IN	T2
0	T3
0	T4

Micro-Instruction for MOVAB.

MOVCA:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << REGC_OUT | 1 << REGA_IN	T2
0	T3
0	T4

Micro-Instruction for MOVCA.

MOVBC:

1 << PC_OUT|1 << MAR_IN	T0
1 << PC_INC | 1 << MEM_OUT|1 << IR_IN	T1
1 << REGB_OUT | 1 << REGC_IN	T2
0	T3
0	T4

Micro-Instruction for MOVBC.

HALT:

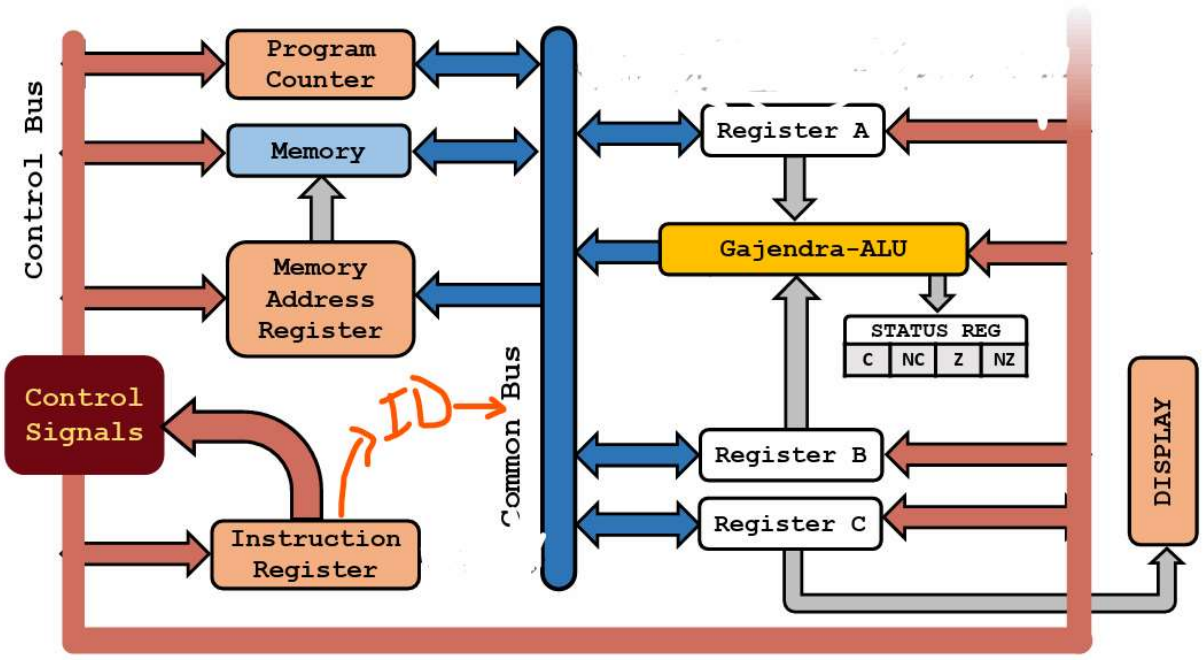
1 << MAR_IN	T0
1 << MEM_OUT|1 << IR_IN	T1
0	T2
0	T3

Micro-Instruction for HALT.

Sample Programs:

The Sample programs which we tried on this Circuit are:

- Add two numbers and displaying the result
  0x15,0x36,0x90,0xf0,0x00,0x34,0x12
- Adding and subtracting four numbers in some combination
  0x15,0x46,0x37,0x48,0xf0,0x17,0x08,0x25,0x12
- Adding numbers from a starting address to ending address and displaying the result
  0x52,0x38,0x39,0x3a,0x3b,0x3c,0x3d,0xf0,0x19,0x26,0x4b,0x04,0x10,0x09
- Multiplication routine using repeated addition
  0x19,0x70,0x50,0x30,0x70,0x4b,0x70,0x83,0xf0,0x16,0x17,0x01


Control Bus
Program Counter
Memory
Memory Address Register
Control Signals
Instruction Register
ID
Common Bus
Register A
Gajendra-ALU
STATUS REG
C
NC
Z
NZ
Register B
Register C
DISPLAY