CS2310: Final Assignment

Computer System Design: Gajendra-I

Teammate1 :-

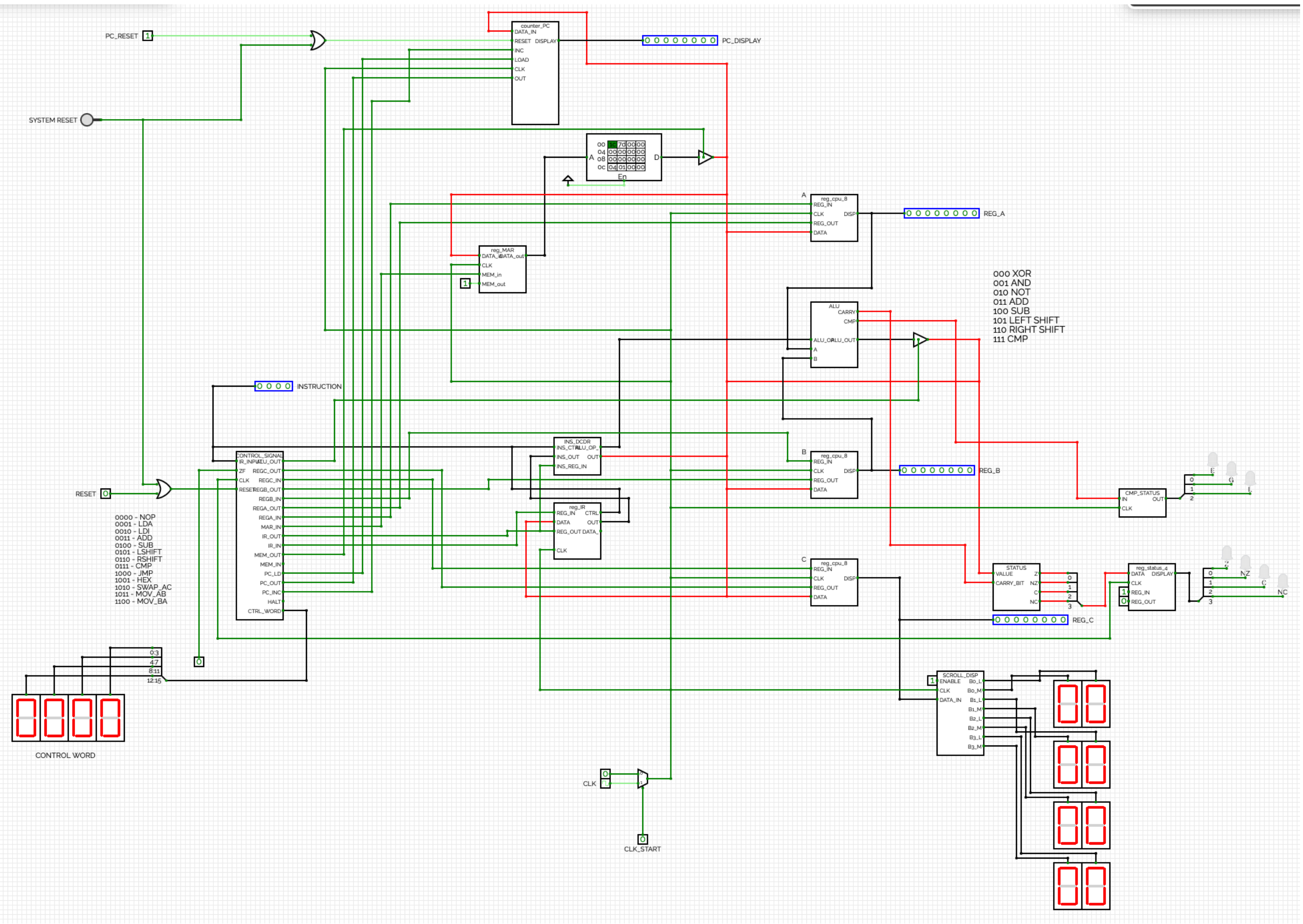
Name:- Prince Garg Roll No. :- CS22B011

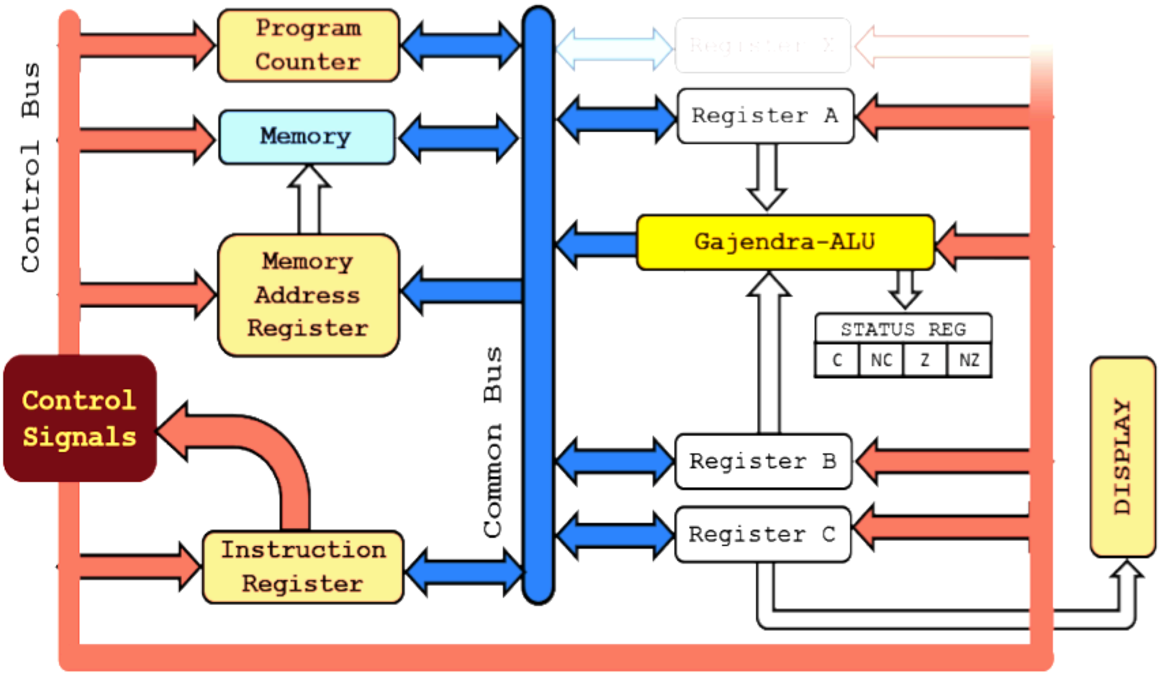
Teammate 2:-

Name:- Vikash Kumar Ojha Roll No. :- CS22B013

Section - 1 - ARCH_GAJENDRA

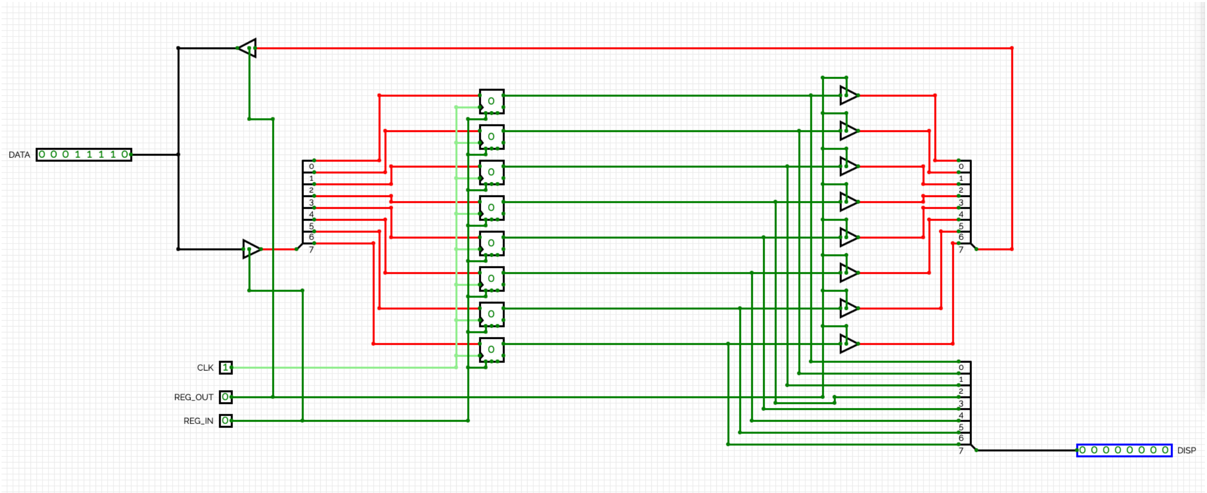
Overall Architecture for CPU core:-



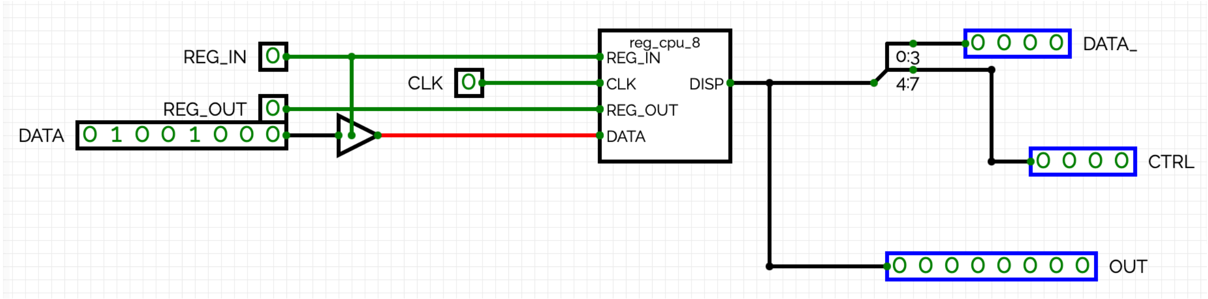


Various internal components used are:-

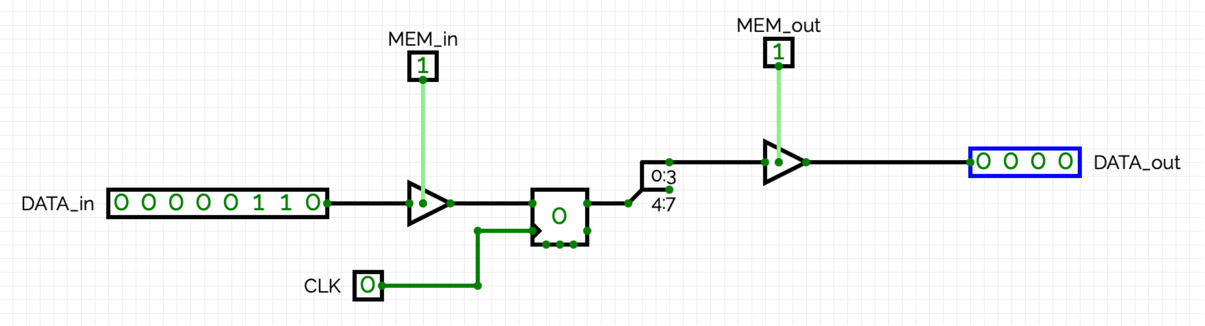
1) General CPU registers:- Stores 8-bit data for dynamic use.



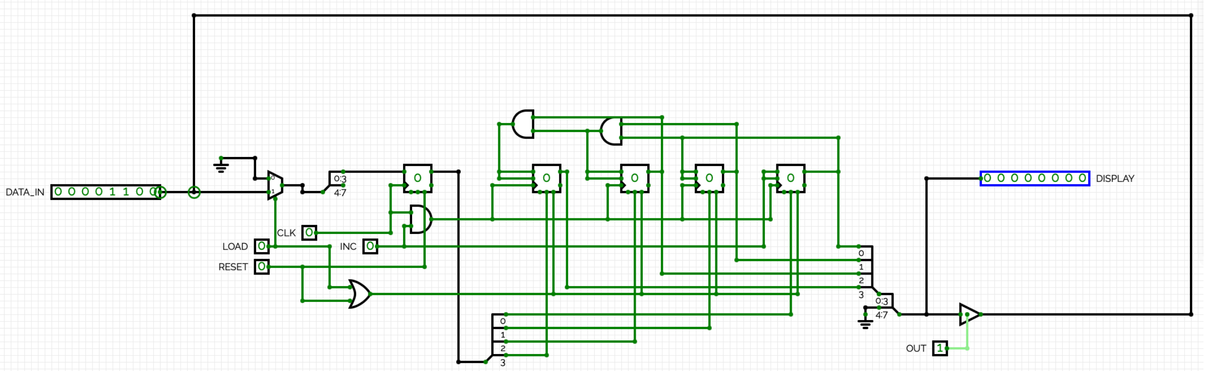
2) Instruction Register:- Separates instruction and data/address passed from the control word.



3) Memory Address Register:- Stores the 4-bit address to be fetched from the ROM.



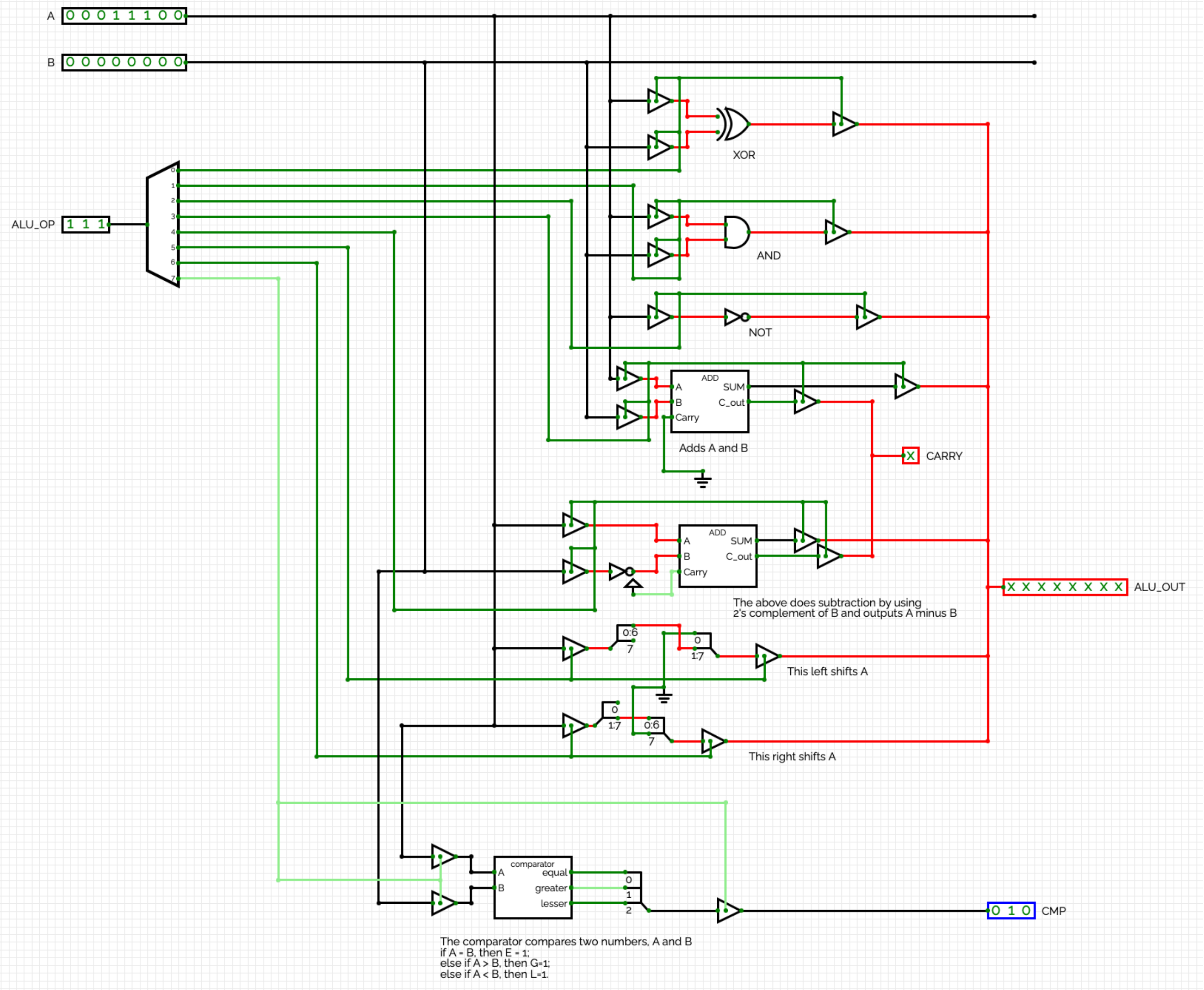
4) Program Counter:- Stores, loads, increments and passes 4 bits indicating the opcode to be fetched from the ROM.



5) Arithmetic And Logical Unit:- Can perform various tasks for different ALU operation codes.

000	-	XOR
001	-	AND
010	-	NOT
011	-	ADD
100	-	SUB
101	-	LEFT SHIFT

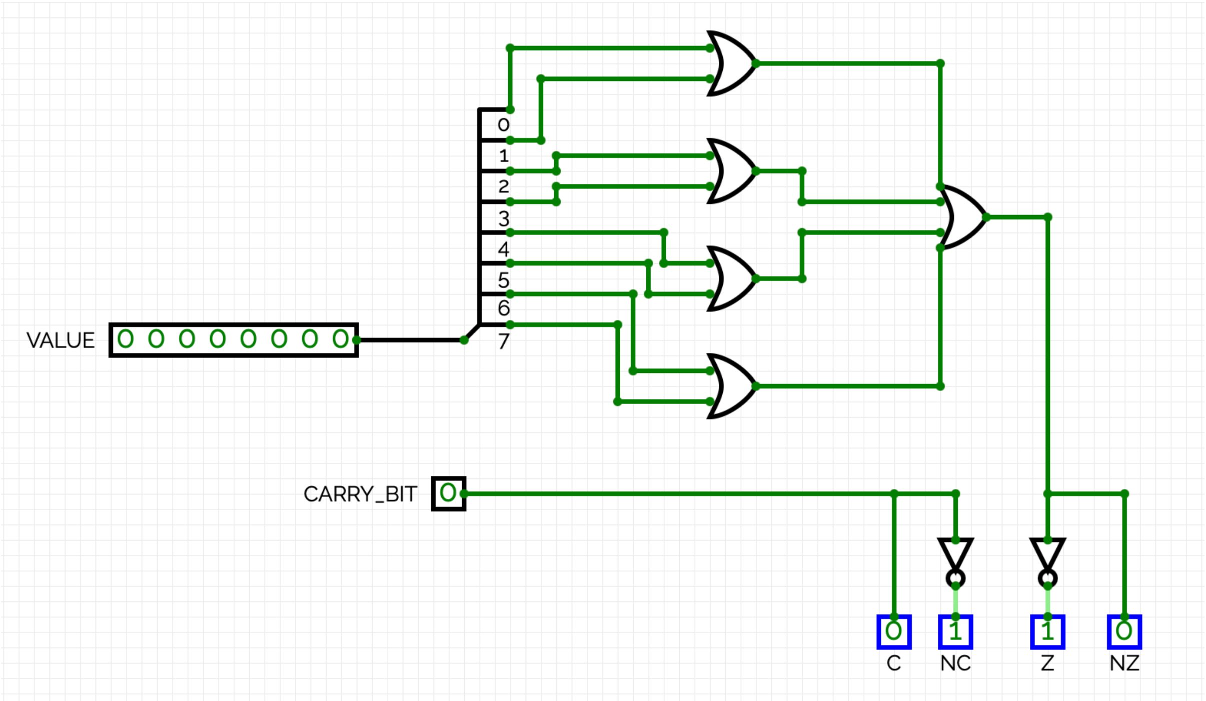
110	-	RIGHT SHIFT
111	-	CMP



6) Status Register:- Gives the status of the output of operation done by ALU in true/false.

States if the bit is zero or nonzero and if the bit has a carry-over or not after MSB.

Status Register:-



Section - 2 - Defining Instruction Set

1) NOP - No Operation

Description:-

Performs single cycle no operation.

Operation:-

No operation

Syntax:-	Operands:-	Program Counter:-
NOP	None	PC <— PC+1

8-bit opcode:-

0000 0000

2) LDA - Load A(Accumulator)

Description:-

Loads the data from a given address into register A.

Operation:-

(i) A <— MEM[Address]

Syntax:-	Operands:-	Program Counter:-
LDA Address	0 <= d <= 15	PC <— PC+1

8-bit opcode:-

0001	dddd

3) LDI - Load Immediate

Description:-

Loads specified data into register A.

Operation:-

(i) A <— DATA

Syntax:-	Operands:-	Program Counter:-
LDI DATA	0 <= d <= 15	PC <— PC+1

8-bit opcode:-

0010	dddd

4) ADD - Addition Without Carry

Description:-

Adds value stored at the given address to register A without using the carry flag.

Operation:-

(i) B <— MEM[Address]

(ii) A <— A + B

Syntax:-	Operands:-	Program Counter:-
ADD Address	0 <= d <= 15	PC <— PC+1

8-bit opcode:-

0011	dddd

5) SUB - Subtraction Without Carry

Description:-

Subtracts value stored at the given address from register A without using the carry flag.

Operation:-

(i) B <— MEM[Address]

(ii) A <— A - B

Syntax:-	Operands:-	Program Counter:-
SUB Address	0 <= d <= 15	PC <— PC+1

8-bit opcode:-

0100	dddd

6) LSHIFT - LEFT SHIFT

Description:-

Left shifts the value stored in register A(Accumulator) stores back to A.

Operation:-

(i) A <— (A << 1)

Syntax:-	Operands:-	Program Counter:-
LSHIFT	None	PC <— PC+1

8-bit opcode:-

0101 0000

7) RSHIFT - RIGHT SHIFT

Description:-

Right shifts the value stored in register A(Accumulator) stores back to A.

Operation:-

(i) A <— (A >> 1)

Syntax:-	Operands:-	Program Counter:-
RSHIFT	None	PC <— PC+1

8-bit opcode:-

0110 0000

8) CMP - COMPARATOR

Description:-

Compares the value stored at the given address with the value stored in register A(Accumulator).

Sets flag:-

L = 1; if A < MEM[Address]

E = 1; if A = MEM[Address]

G = 1; if A > MEM[Address]

Operation:-

(i) B <— MEM[Address]

(ii) Compare and Set Flag

Syntax:-	Operands:-	Program Counter:-
CMP Address	0 <= d <= 15	PC <— PC+1

8-bit opcode:-

0111	dddd

9) JMP - UNCONDITIONAL JUMP

Description:-

Changes the value of the program counter to the value specified.

Operation:-

(i) PC <— ADDRESS

Syntax:-	Operands:-	Program Counter:-
JMP ADDRESS	0 <= d <= 15	PC <— ADDRESS

8-bit opcode:-

1000	dddd

10) HEX - DISPLAY OUTPUT

Description:-

Displays the value stored in register A as output via register C.

Operation:-

(i) C <— A

Syntax:-	Operands:-	Program Counter:-
HEX	None	PC <— PC+1

8-bit opcode:-

1001	0000

11) SWAP_AC -

Description:-

Swaps values of register A(Accumulator) and register C.

Operation:-

(i) B <— A

(ii) A <— C

(iii) C <— B

Syntax:-	Operands:-	Program Counter:-
SWAP_AC	None	PC <— PC+1

8-bit opcode:-

1010	0000

12) MOV_AB - MOVE A to B

Description:-

Moves value in register A(Accumulator) to register B.

Operation:-

(i) B <— A

Syntax:-	Operands:-	Program Counter:-
MOV_AB	None	PC <— PC+1

8-bit opcode:-

1011	0000

13) MOV_BA - MOVE B to A

Description:-

Moves value in register B to register A(Accumulator).

Operation:-

(i) A <— B

Syntax:-	Operands:-	Program Counter:-
MOV_BA	None	PC <— PC+1

8-bit opcode:-

1100	0000

Instruction Set:-

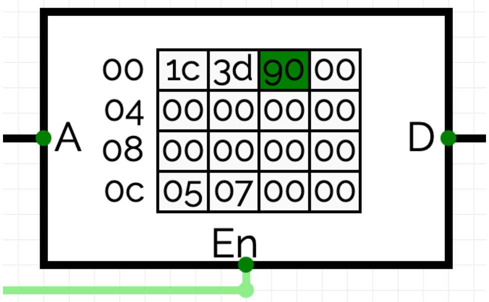
0000	-	NOP
0001	-	LDA
0010	-	LDI
0011	-	ADD
0100	-	SUB
0101	-	LSHIFT
0110	-	RSHIFT
0111	-	CMP
1000	-	JMP
1001	-	HEX
1010	-	SWAP_AC
1011	-	MOV_AB
1100	-	MOV_BA

Section - 3 - Assembly Programs Examples

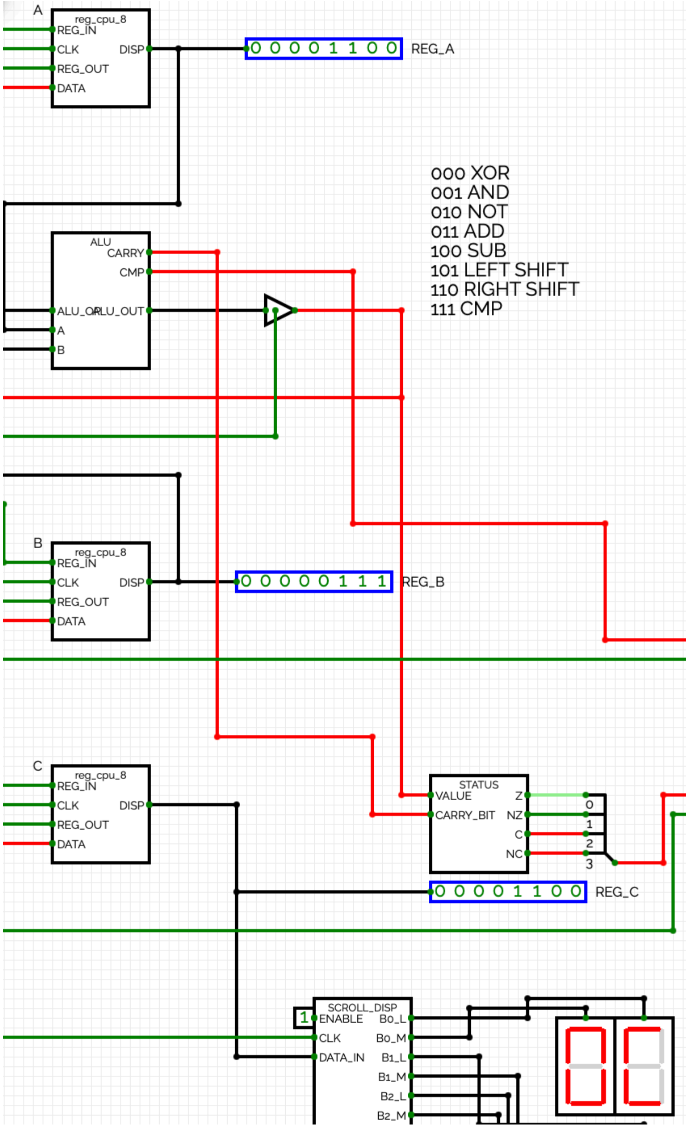
1. Adding two numbers and displaying the result

Adding numbers stored at address 0xC and 0xD.
Answer=0x05+0x07=0x0C=>00001100

Control Rom:- Registers:-



00	1c	3d	90	00
04	00	00	00	00
08	00	00	00	00
0c	05	07	00	00



000 XOR
001 AND
010 NOT
011 ADD
100 SUB
101 LEFT SHIFT
110 RIGHT SHIFT
111 CMP

OUTPUT :- register C = 0x0C

PROGRAM:-

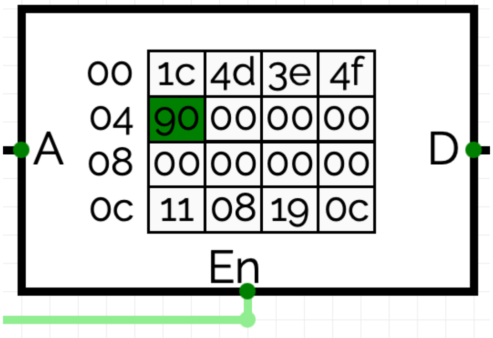
```
0x0    LDA   0xC
0x1    ADD   0xD
0x2    HEX
```

2. Adding and subtracting four numbers in some combination (e.g., 17-8+25-12)

Stored
17, i.e., 0x11 at 0xC
8, i.e., 0.08 at 0xD
25, i.e., 0x19 at 0xE
12, i.e., 0x0C at 0xF

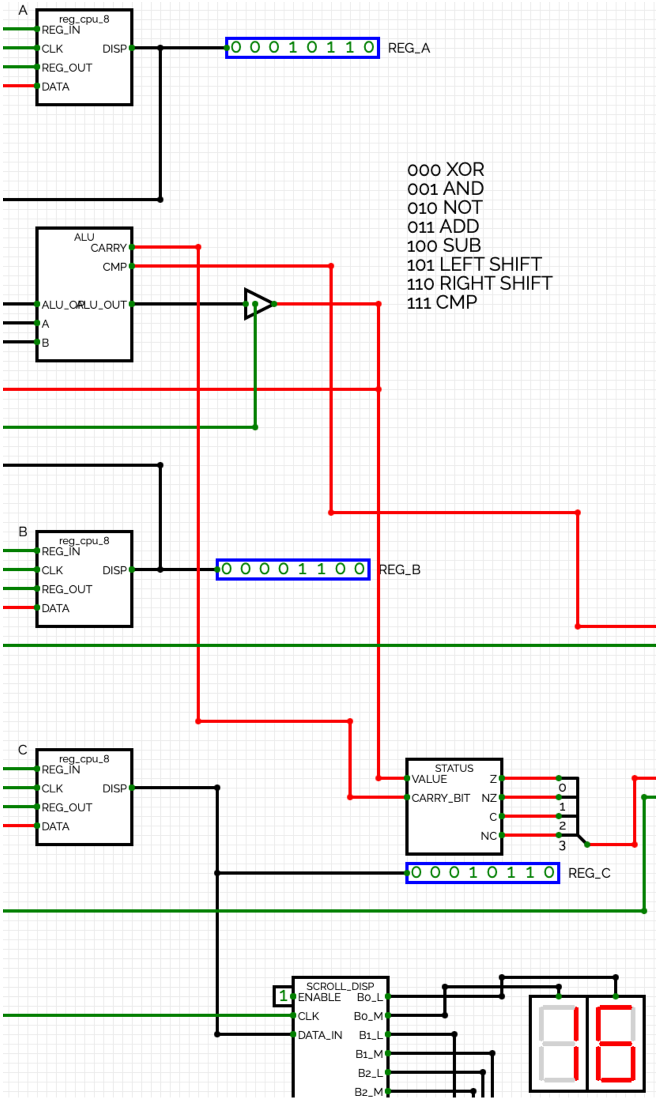
Answer = 17-8+25-12 = 22 = 0x16

Control ROM:-



00	1c	4d	3e	4f
04	90	00	00	00
08	00	00	00	00
0c	11	08	19	0c

Registers:-



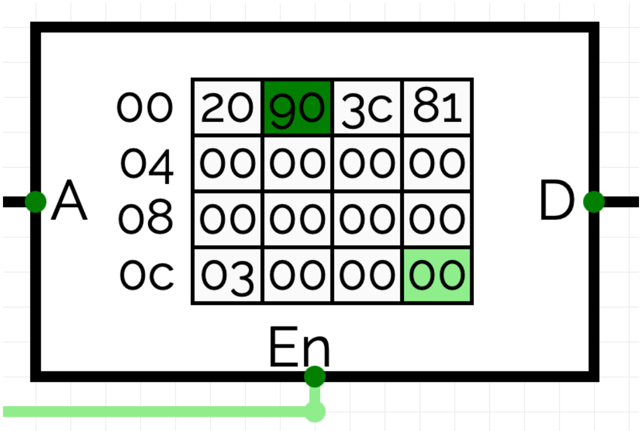
OUTPUT :- register C = 0x16

PROGRAM:-

```
0x0    LDA    0xC
0x1    SUB    0xD
0x2    ADD    0xE
0x3    SUB    0xF
0x4    HEX
```

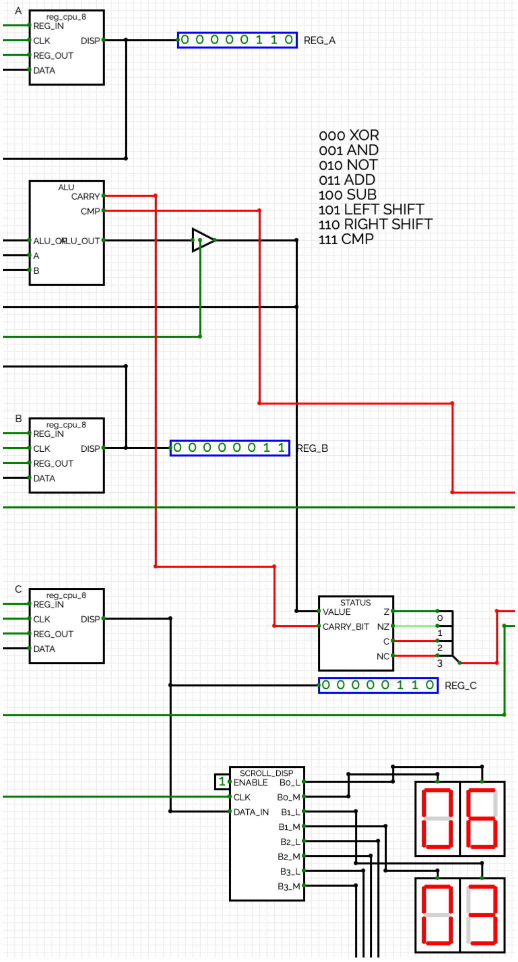
3. Display multiplication table for a number stored at some address.

Control ROM :-



00	20	90	3c	81
04	00	00	00	00
08	00	00	00	00
0c	03	00	00	00

Registers:-



Output:- Register C:-

first it displays 0x00

next cycle :- 0x03

next :- 0x06

next :- 0x09

and so on…

PROGRAM:-

```
0x0    LDI    0x0
0x1    HEX
0x2    ADD    0xC
0x3    JMP    0x1
```

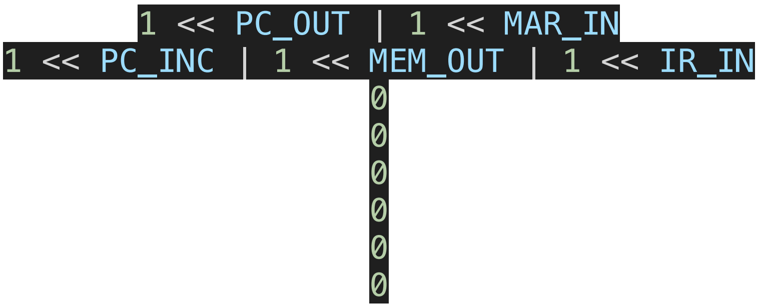
Section - 4 - Microinstructions and controller logic design

	PC_INC	PC_OUT	PC_LOAD	MEM_IN	MEM_OUT	IR_IN	IR_OUT	MAR_IN	REGA_IN	REGA_OUT	REGB_IN	REGB_OUT	REGC_IN	REGC_OUT
T_0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
T_1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
T_2														

Column Numbers:-

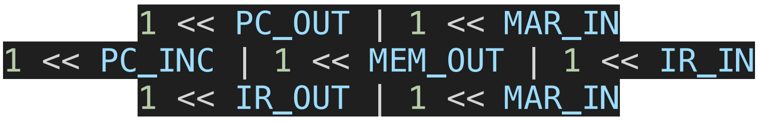
PC_INC = 14
PC_OUT = 13
PC_LOAD = 12
MEM_IN = 11
MEM_OUT = 10
IR_IN = 9
IR_OUT = 8
MAR_IN = 7
REGA_IN = 6
REGA_OUT = 5
REGB_IN = 4
REGB_OUT = 3
REGC_IN = 2
REGC_OUT = 1
ALU_OUT = 0

NOP:- 0000

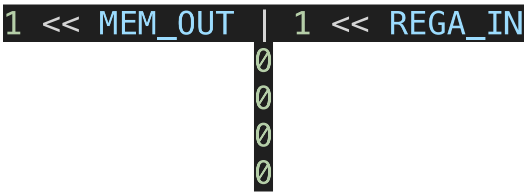


```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
0
0
0
0
0
0
```

LDA:- 0001

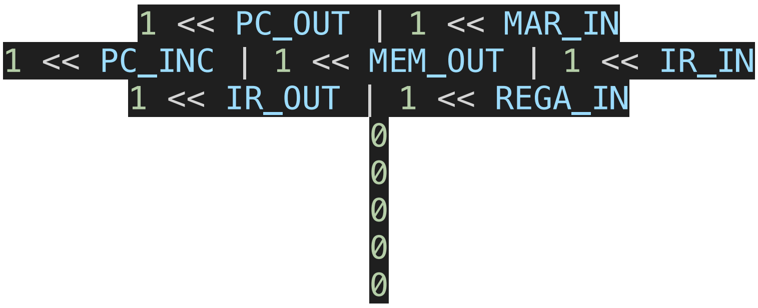


```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << IR_OUT | 1 << MAR_IN
```



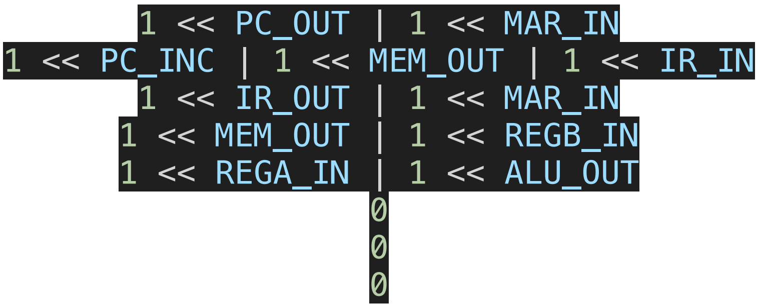
```
1 << MEM_OUT | 1 << REGA_IN
0
0
0
0
```

LDI:- 0010



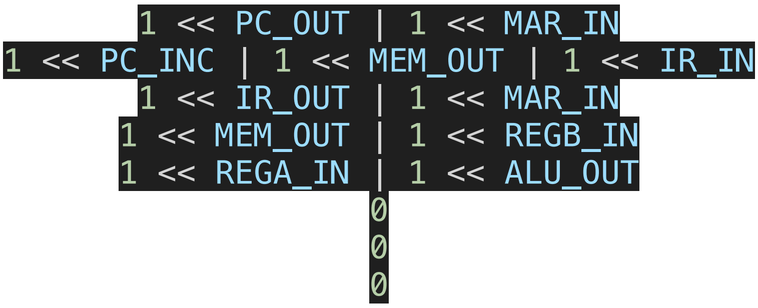
```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << IR_OUT | 1 << REGA_IN
0
0
0
0
0
```

ADD:- 0011



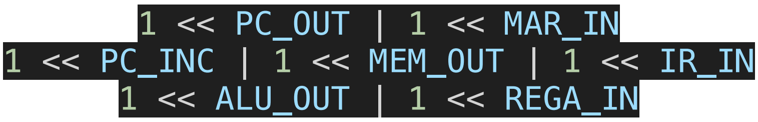
```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << IR_OUT | 1 << MAR_IN
1 << MEM_OUT | 1 << REGB_IN
1 << REGA_IN | 1 << ALU_OUT
0
0
0
```

SUB:- 0100

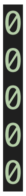


```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << IR_OUT | 1 << MAR_IN
1 << MEM_OUT | 1 << REGB_IN
1 << REGA_IN | 1 << ALU_OUT
0
0
0
```

LSHIFT:- 0101

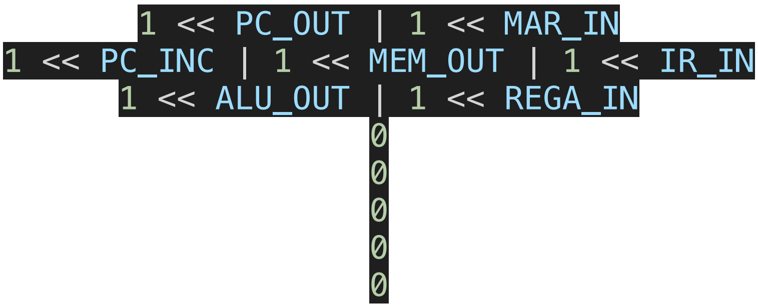


```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << ALU_OUT | 1 << REGA_IN
```



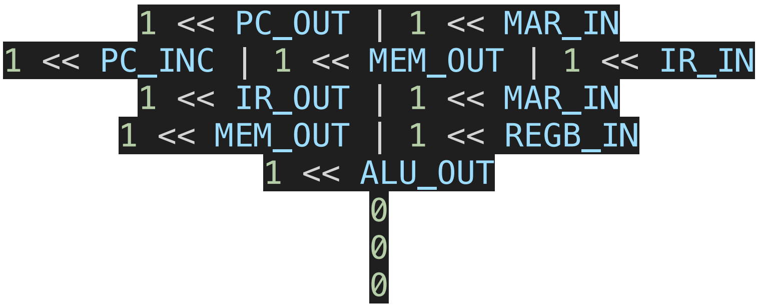
```
0
0
0
0
0
```

RSHIFT:- 0110



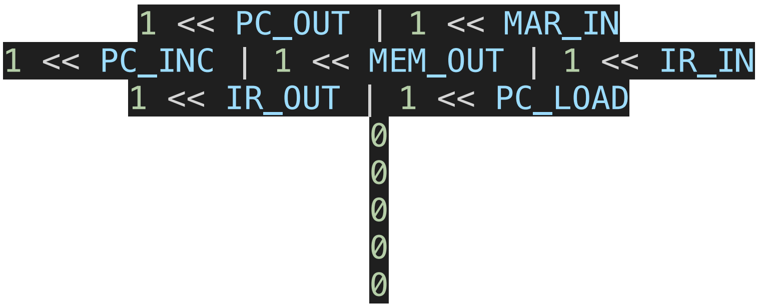
```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << ALU_OUT | 1 << REGA_IN
0
0
0
0
0
```

CMP:- 0111



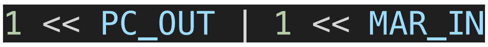
```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << IR_OUT | 1 << MAR_IN
1 << MEM_OUT | 1 << REGB_IN
1 << ALU_OUT
0
0
0
```

JMP:- 1000

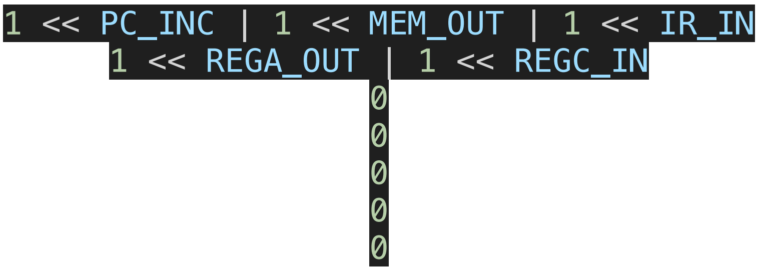


```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << IR_OUT | 1 << PC_LOAD
0
0
0
0
0
```

HEX:- 1001

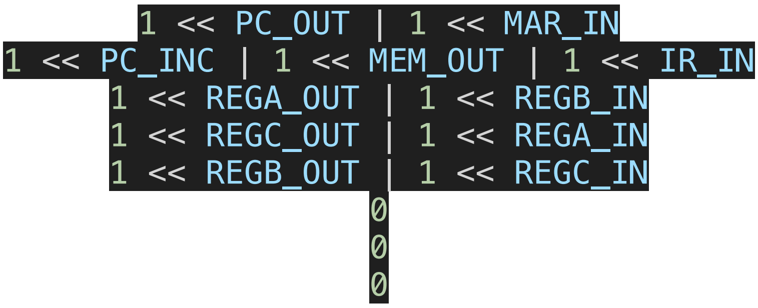


```
1 << PC_OUT | 1 << MAR_IN
```



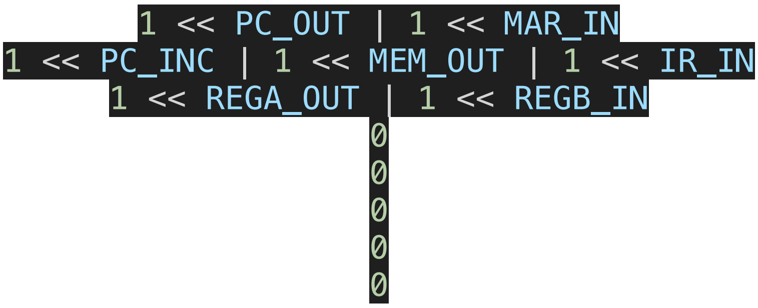
```
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << REGA_OUT | 1 << REGC_IN
0
0
0
0
0
```

SWAP_AC:- 1010



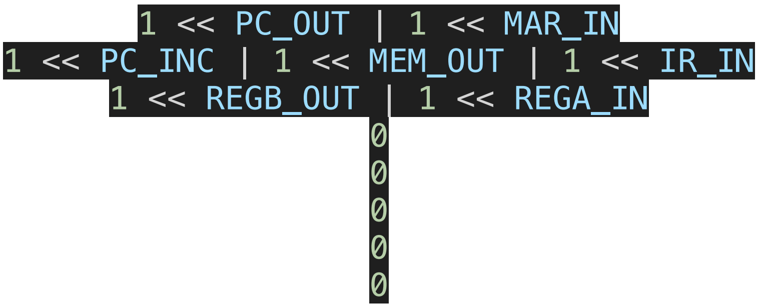
```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << REGA_OUT | 1 << REGB_IN
1 << REGC_OUT | 1 << REGA_IN
1 << REGB_OUT | 1 << REGC_IN
0
0
0
```

MOV_AB:- 1011



```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << REGA_OUT | 1 << REGB_IN
0
0
0
0
0
```

MOV_BA:- 1100

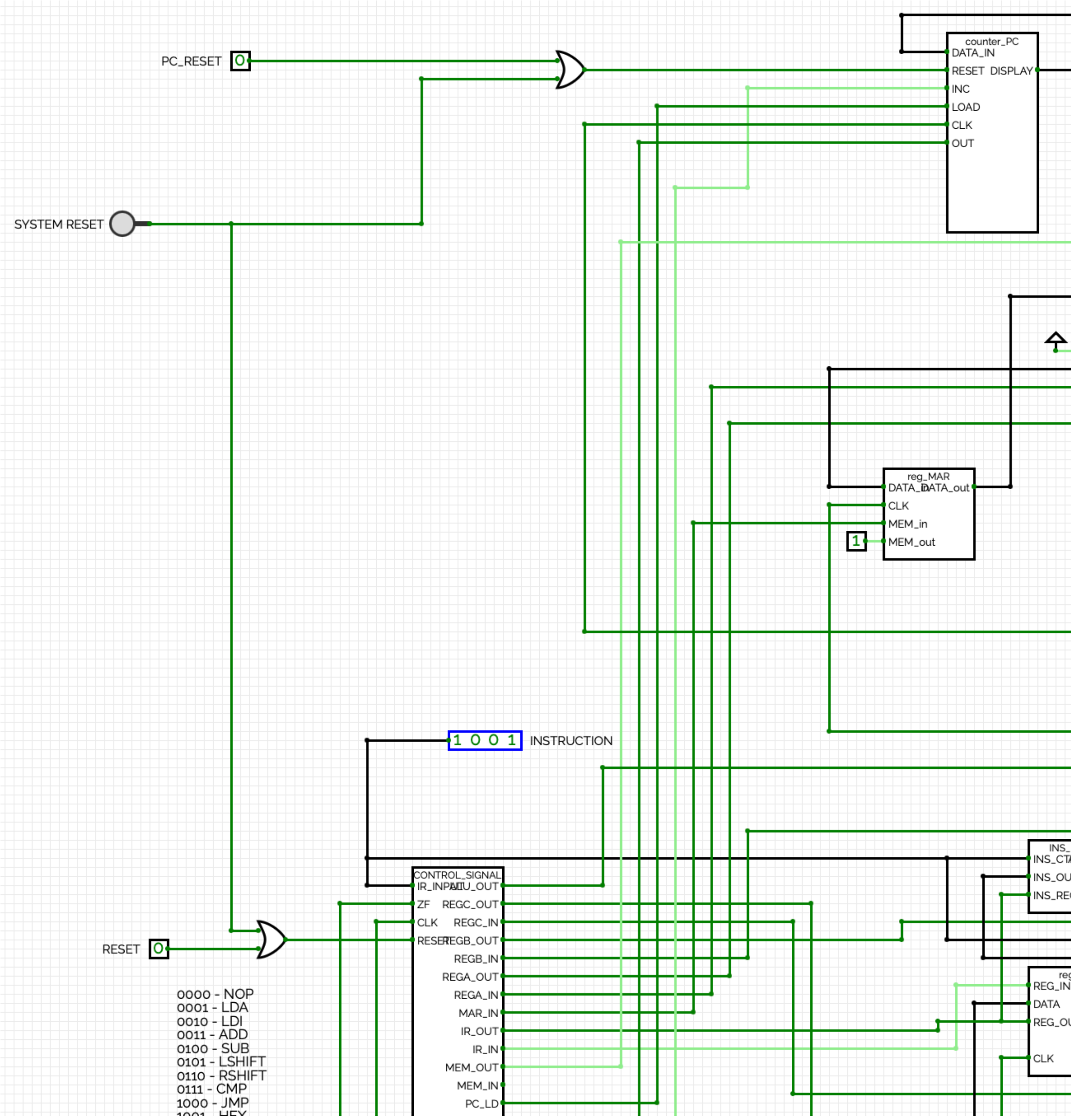


```
1 << PC_OUT | 1 << MAR_IN
1 << PC_INC | 1 << MEM_OUT | 1 << IR_IN
1 << REGB_OUT | 1 << REGA_IN
0
0
0
0
0
```

The minimum number of T-states needed for this instruction set is 5. However, we have used 8 T-states for each instruction.

Section - 5 - System Reset

System Reset:- Reset button that reboots the computer to execute instructions from address 0x0. It also resets the PC count to zero and the controller's internal counter.



Additional (extra credits) Extension

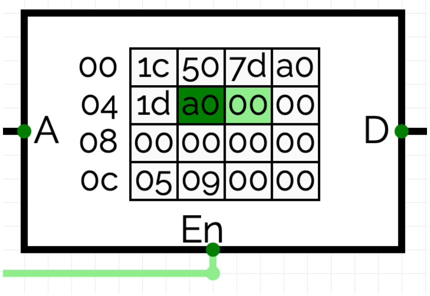
1)[Simple] Update the ALU to support instructions like CMP (compare), SWAP, some MOV instructions, SHIFT (Left/Right) etc. A program must be written to demonstrate the capability of such instructions.

—> Updated the ALU to support CMP, SWAP_AC, MOV_AB, MOV_BA, LSHIFT and RSHIFT functions also.

—> Wrote C program for the instruction set and updated the EEPROM to contain the instructions.

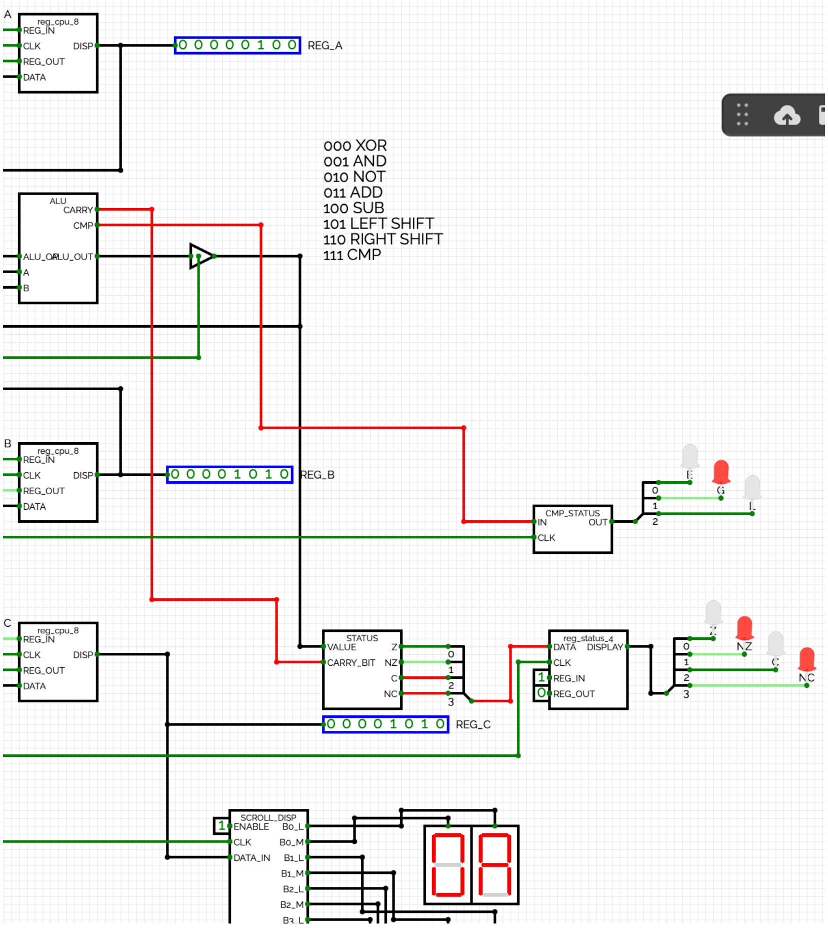
Ques:- Fetch two numbers stored in memory, left shift one, compare them and print one by one.

Control ROM :-



00	1c	50	7d	a0
04	1d	a0	00	00
08	00	00	00	00
0c	05	09	00	00

Registers:-



000 XOR
001 AND
010 NOT
011 ADD
100 SUB
101 LEFT SHIFT
110 RIGHT SHIFT
111 CMP

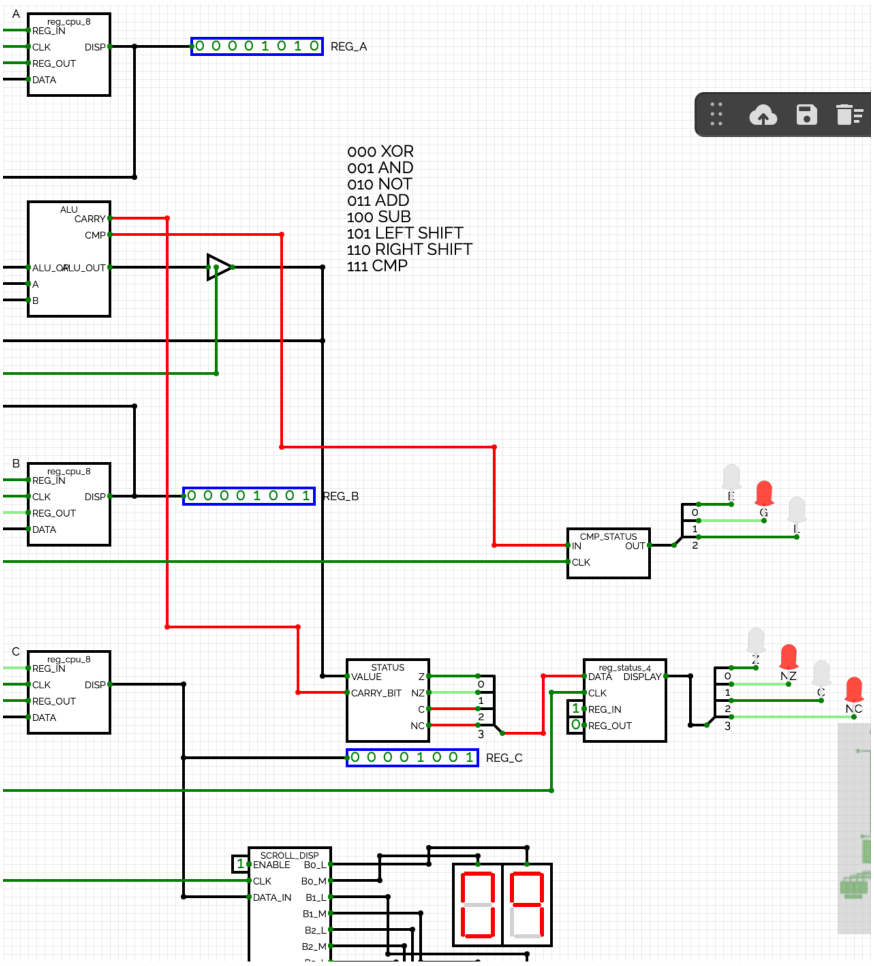
OUTPUT:-

left shift of 0x05 = 0x0A

0x0A > 0x09 hence, G flag is set to 1.

PROGRAM:-

```
0x0    LDA        0xC
0x1    LSHIFT
0x2    CMP        0xD
0x3    SWAP_AC
0x4    LDA        0xD
0x5    SWAP_AC
```



000 XOR
001 AND
010 NOT
011 ADD
100 SUB
101 LEFT SHIFT
110 RIGHT SHIFT
111 CMP