CS2310 Final Project

GAJENDRA CPU

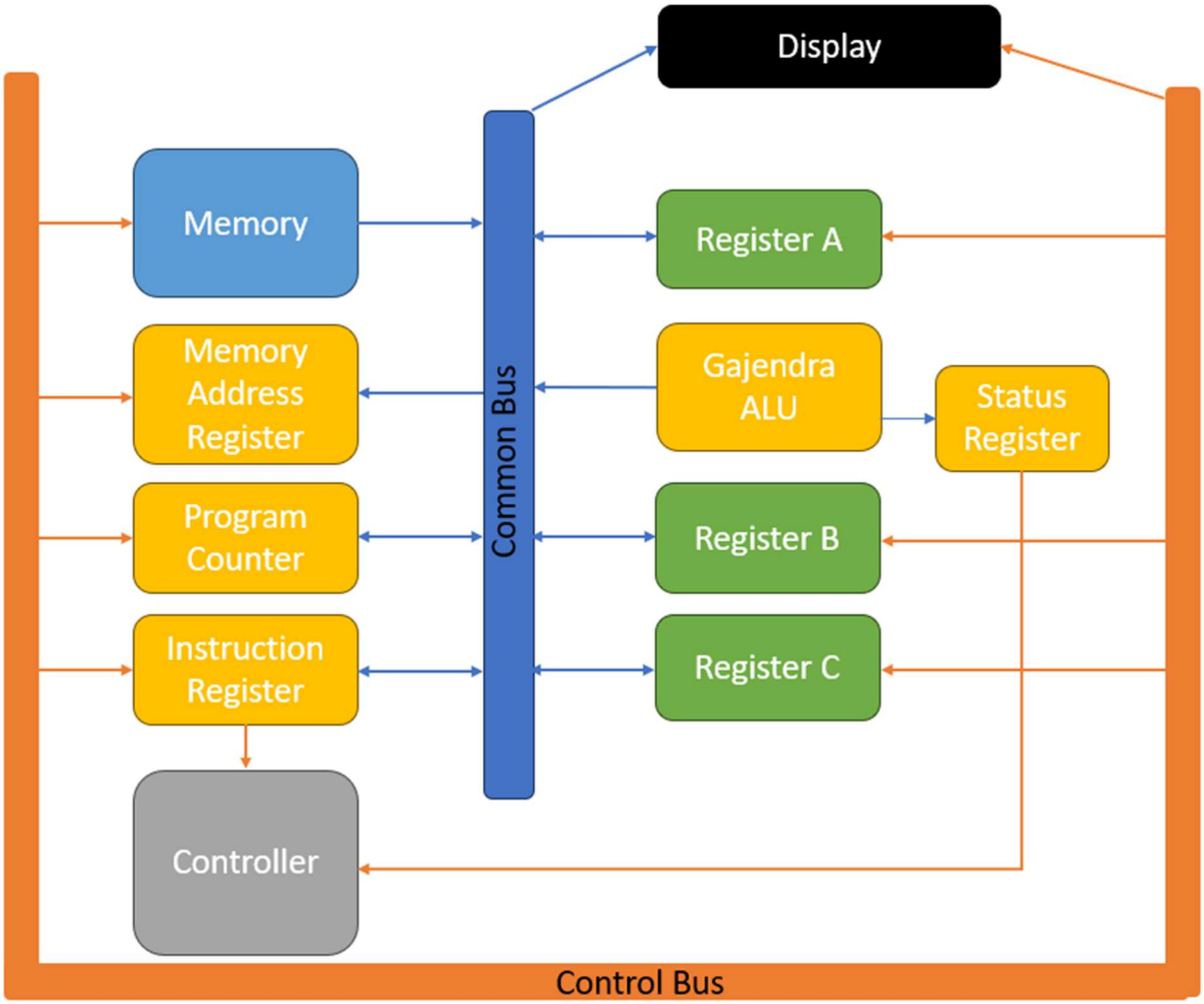
❖ Contents :

- Overall Architecture of CPU
- Instruction Set Descriptions
- Few programs implemented using the CPU
- Extra Credit

- Raadhes Chandaluru (CS22B069)
- Yashvardhan Toshniwal (CS22B088)
- Aditya Srivastava (CS22B066)

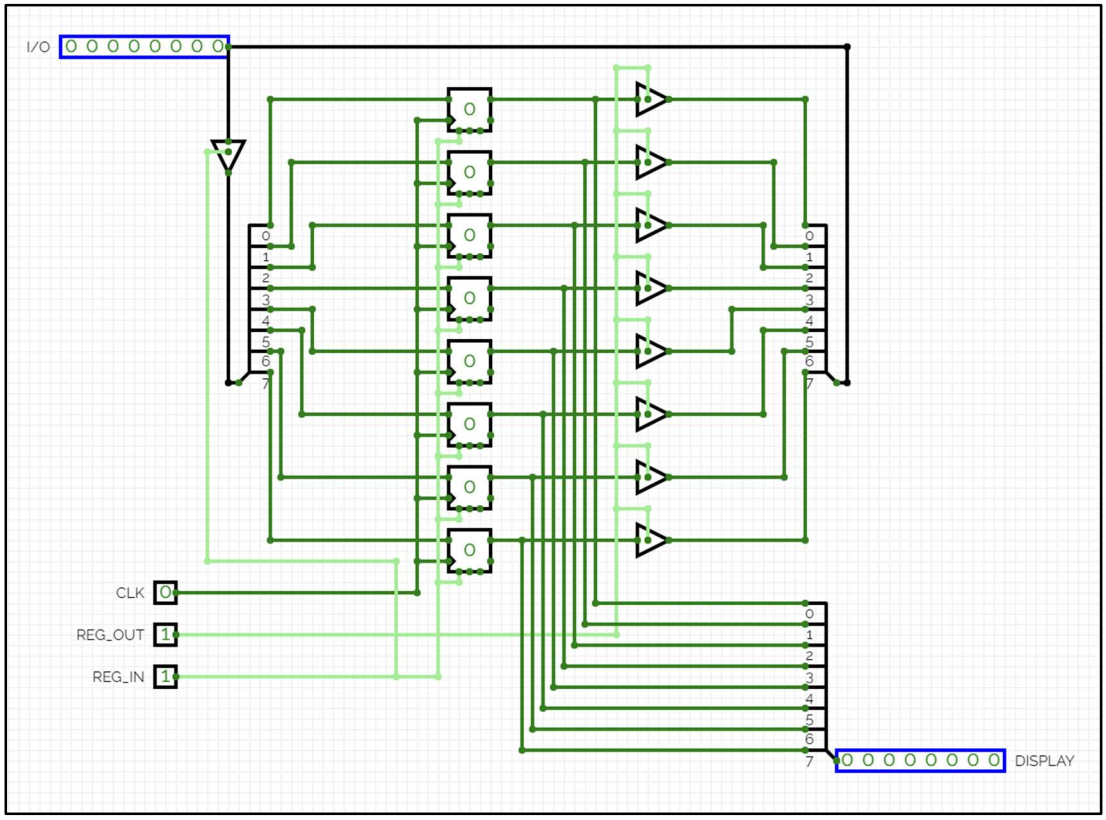
Overall Architecture of CPU

ARCH_GAJENDRA



1. General CPU Registers (`reg_cpu_8`)

I. The General CPU register module is used as a general purpose register to store 8 bit data / values . Registers A (Accumulator) , Register B and Register C are `reg_cpu_8` modules .



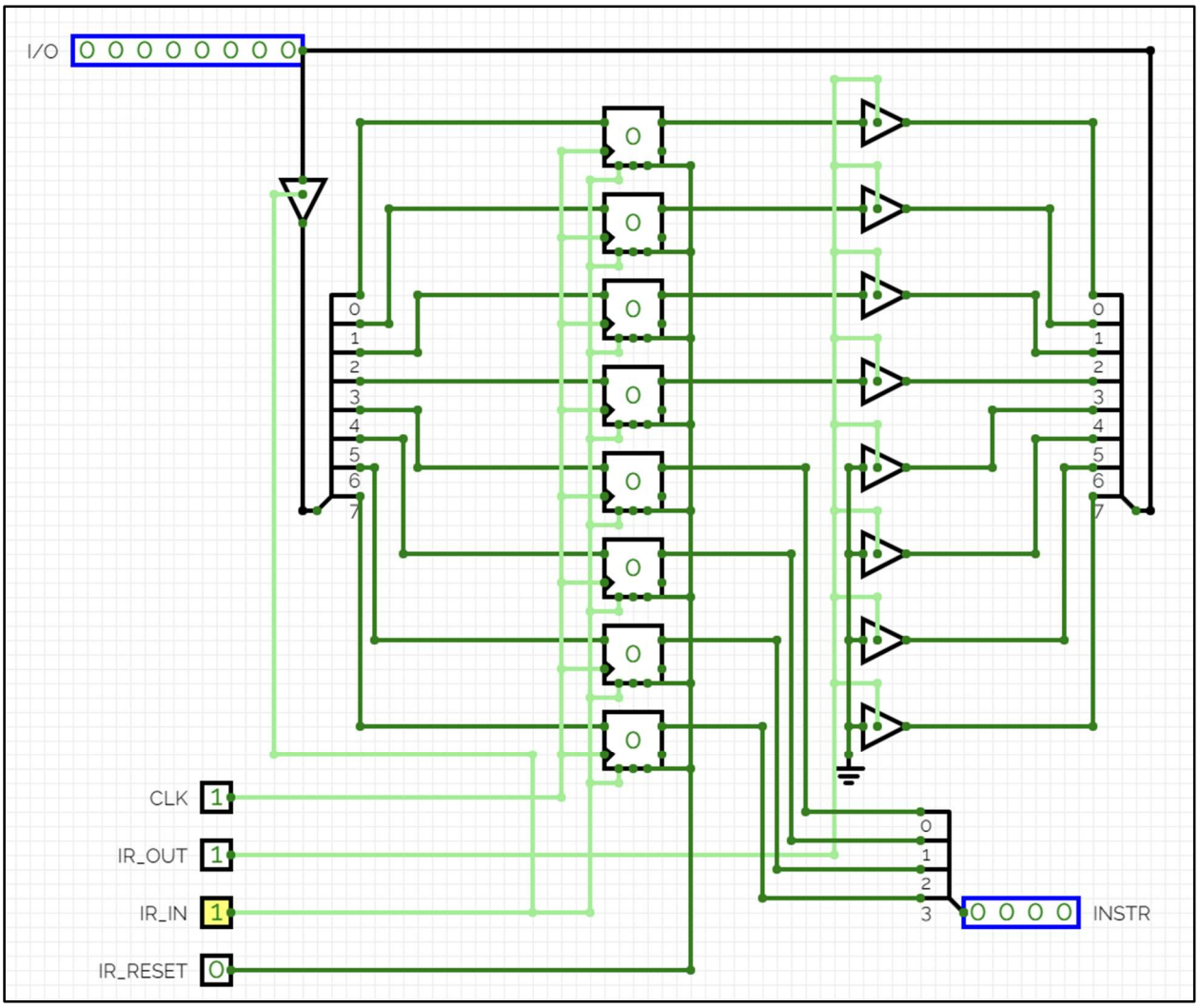
II. **Design** : There is a common Input/Output stream (**I/O**) , Display output stream (**DISPLAY**) , Clock input (**CLK**) , and control signals **REG_OUT**, **REG_IN** .

III. When REG_IN is high , the register stores 8 - bit value of I/O stream in the D - flip flops at the positive clock edge. When REG_OUT is high the register outputs the 8 - bit value stored in the D - flip flops into the I/O stream .

IV. In the CPU core architecture the I/O stream is connected to the common bus .

V. The DISPLAY output stream always outputs the 8 - bit value which is stored in the D - flip flops .

2. Instruction Register (`reg_IR`)



I. Instruction register stores the 8 - bit opcode that specifies the operation the cpu must perform in the instruction cycle .

II. **Design** : There is a common Input/Output stream (**I/O**) , Control output stream (**INSTR**), Clock input (**CLK**) , and control signals **IR_OUT**, **IR_IN, IR_RESET**.

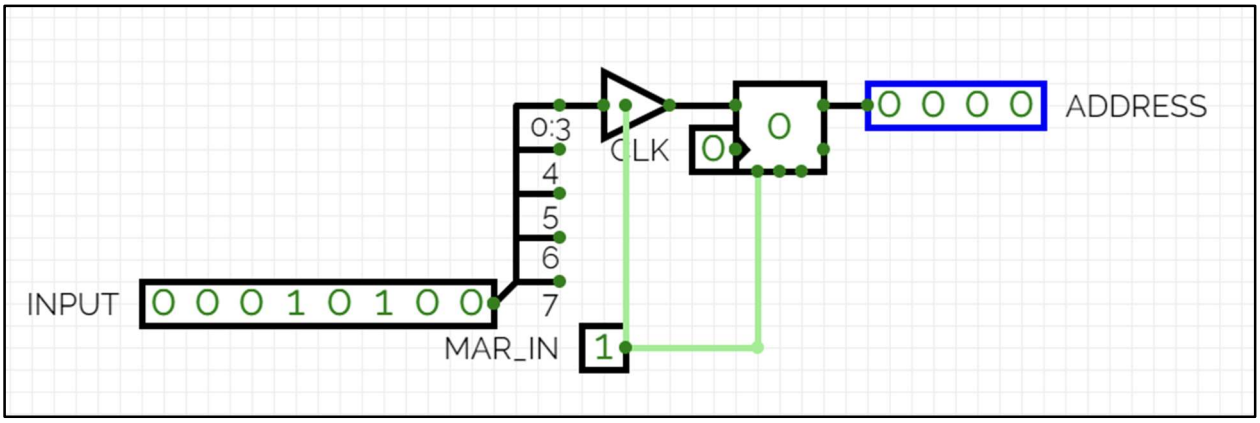
III. The Instruction Register (`reg_IR`) receives input from the I/O stream which is connected to the common bus in CPU architecture and stores it in 8 1-bit D flip-flops at the positive clock edge when IR_IN is set.
The most significant 4 bits of the input represent the machine code for the instruction, while the least significant 4 bits are outputted to the common bus.

IV. When IR_OUT is set ,it is noteworthy that the output fed into the common bus is 8 bits long, with the least significant four bits identical to the least significant 4 bits of the input, and the most significant four bits set to zero. This is particularly crucial for instructions such as LDI.

Example : LDI 5 , while loading register with 5 we must only output the least significant 4 bits

V. The most significant 4 bits are output via **INSTR** output which is fed directly to the control module for logical decoding .

3. Memory Address Register (`reg_MAR`) - 4-bit addresses

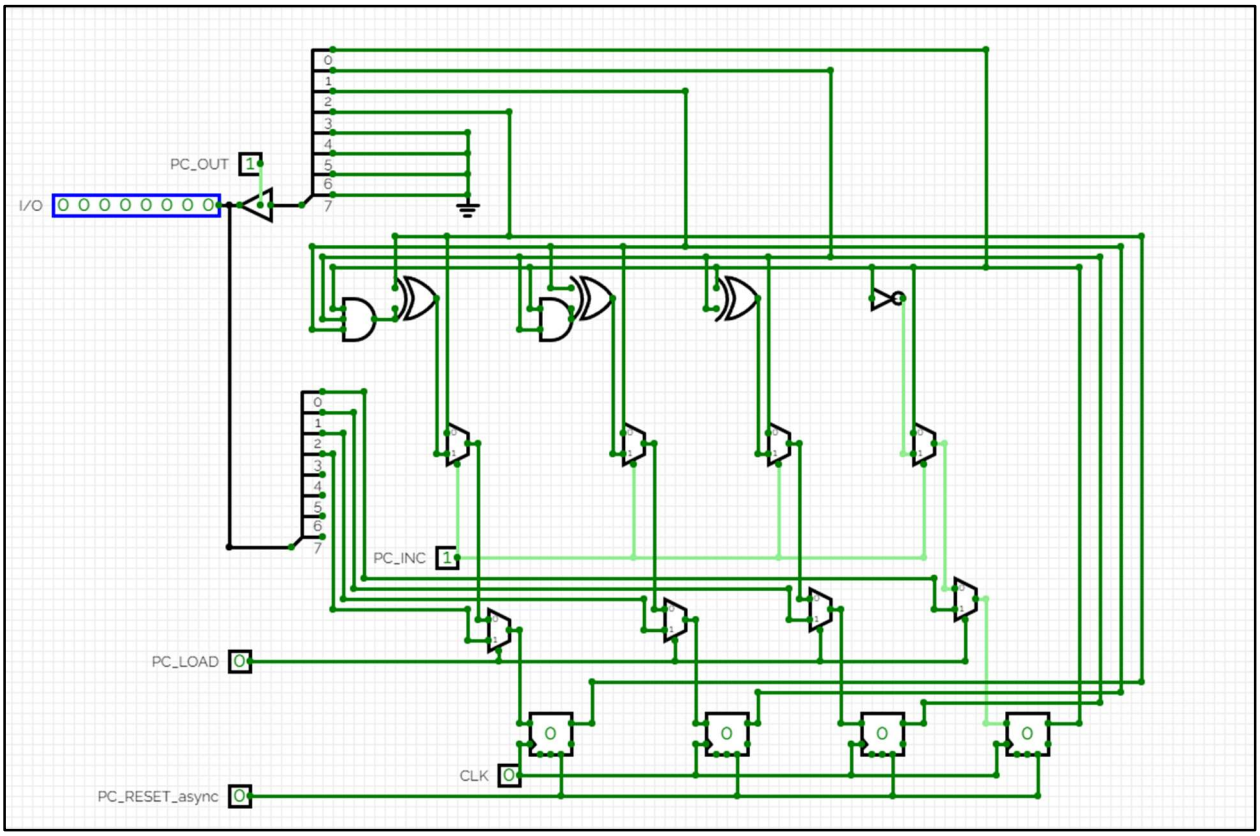


I. The Memory address Register stores the 4 - bit memory address from which data or instructions from the memory module need to be retrieved .

II. **Design** : There is an Input stream (**INPUT**) , Output stream (**ADDRESS**) , Clock input (**CLK**) , and control signals **MAR_IN**.

III. The Memory Address Register (`reg_MAR`) takes input from the common bus. When **MAR_IN** is set, the least significant 4 bits of the input are stored in the memory address register at the rising clock edge, which is then directly fed as the memory address input to the Memory Module .

4. Program Counter (`counter_PC`) - 4-bit addresses



I. The Program Counter keeps a track of the memory address of the instruction which is to be executed .

II. **Design** : There is a common **I/O** stream . The control signals are **PC_RESET_async** , **PC_LOAD** , **PC_INC** , **PC_OUT** . There is also an input Clock(**CLK**)

III. PC_RESET_async when set will immediately reset the 4 - bit memory address stored to 0

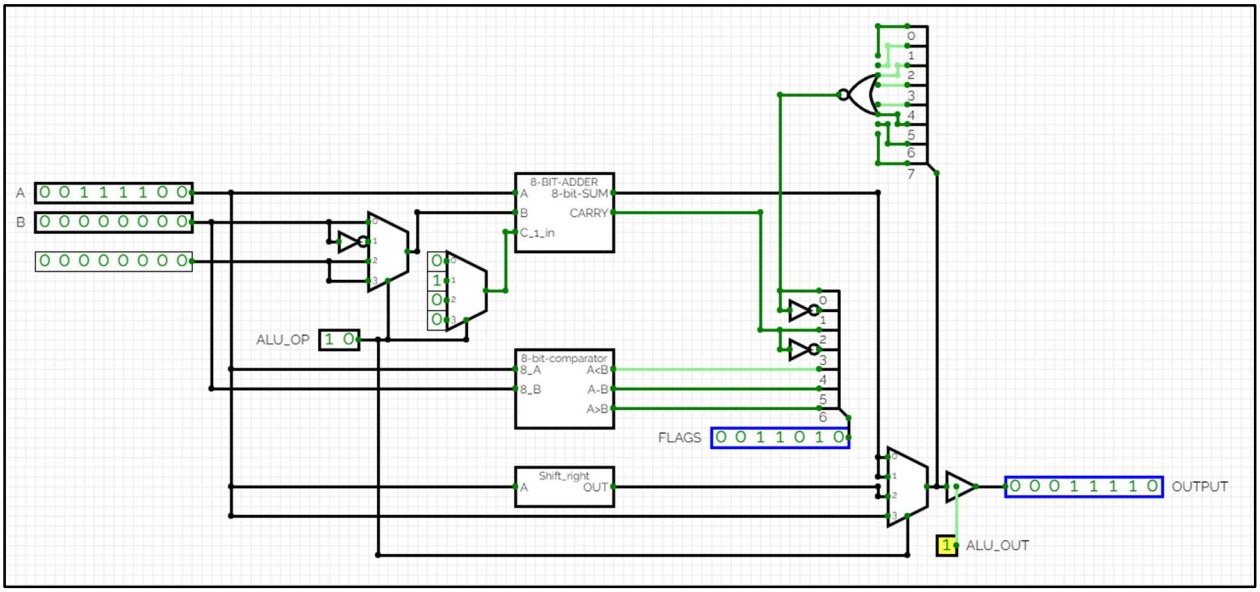
IV. PC_OUT when set will output an 8 - bit value whose 4 most significant bits are 0 and least significant 4 bits are the 4 - bit memory addresses stored in the program counter .

V. PC_LOAD when set to high will allow us to load a 4- bit memory address from the least significant 4- bits of the I/O stream which is connected to the common bus into the program counter internal D - flip flops . This is useful for JMP / JNZ instructions .

VI. PC_INC when set will increment the 4 - bit memory address stored in the program counter by one . This is used in going through the instructions in the memory sequentially .

VII. **Note** : These should normally be no case where PC_LOAD and PC_INC are set at the same time .However if that occurs then the functionality of PC_LOAD is processed by combinational logic implemented .

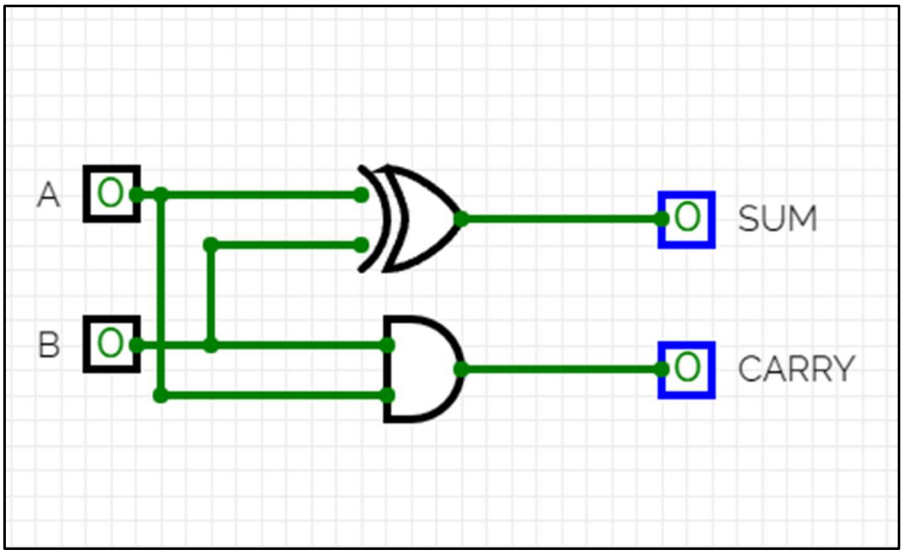
5. Arithmetic and Logical Unit (ALU) - supports only ADD , SUB , SHIFT(right) , CMP operations



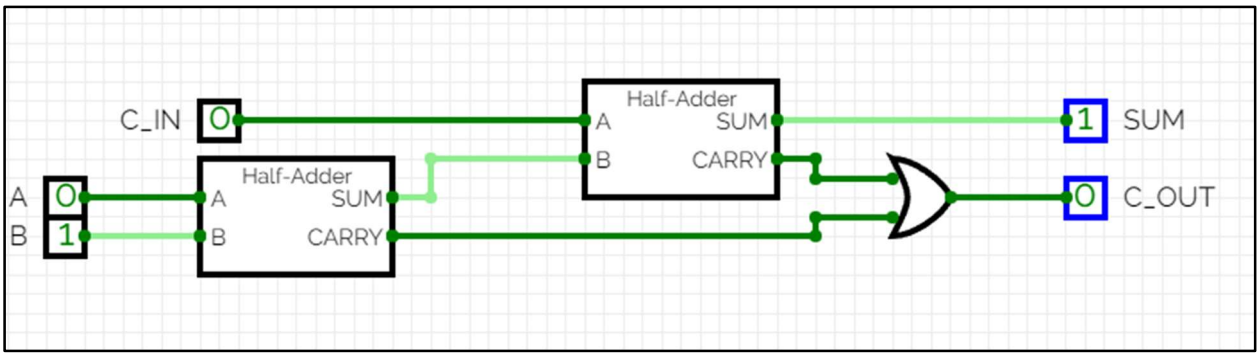
I. The ALU can perform operations such as unsigned addition , unsigned subtraction , shift right by one bit on 8 - bit values
II. It also has a purely comparison based functionality as well
III. **Design** : The inputs to the ALU are 8 - bit values which are in registers **A** , **B** which are connected directly to the ALU in the CPU architecture . The control signals for the ALU are **ALU_OP** which is 2 bits long specifying which functionality to apply , and **ALU_OUT** which controls when the output of the ALU should be output to the common bus .

IV. The ALU also outputs the various flags to a separate 7 - bit output (**FLAGS**) which is directly connected to the status register in the CPU architecture .
V. **Flags** : The Various flags are listed under the status register below .
VI. The **Zero (Z)** flag is set by combinational logic by taking NOR of all bits of result
VII. The **Carry (C)** flag is set by the 8 - bit adder
VIII. The 8 - bit comparator sets the Greater than (**G**) , equal to (**E**) and less than (**L**) flags
IX. 8 - bit Adder is formed from 8 Full adders , where each full adder is formed from 2 half adders

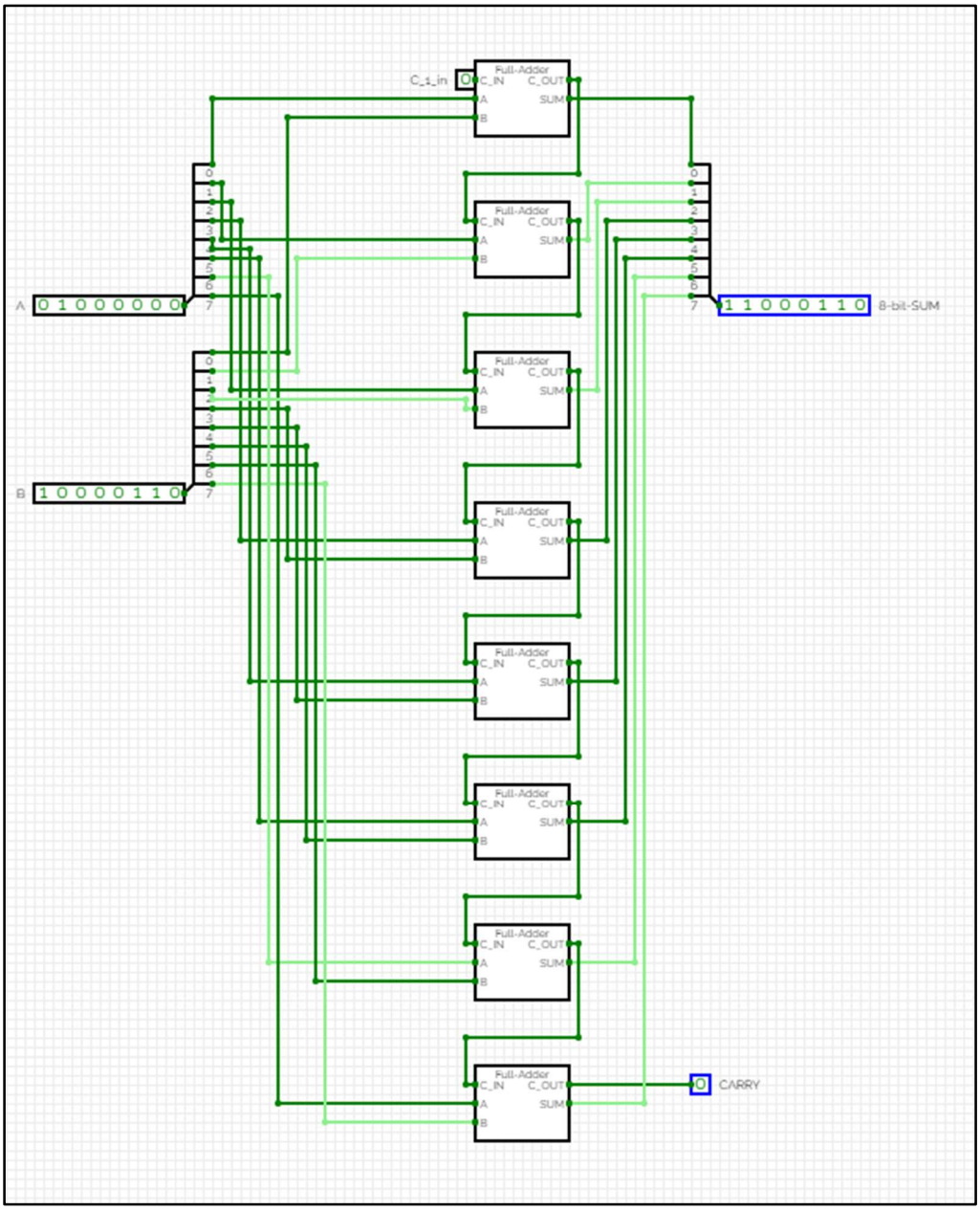
X. Half-Adder :



XI. Full-Adder



XII. 8-bit-Adder

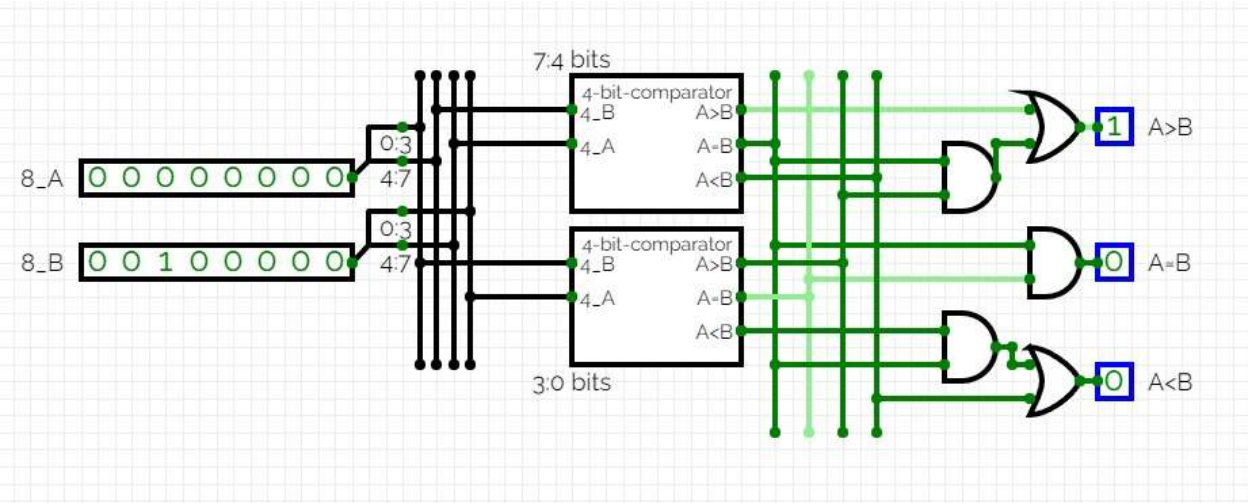


XIII. ALU_OP control implemented via 4:1 MUXs in ALU , the functionality is tabulated below :

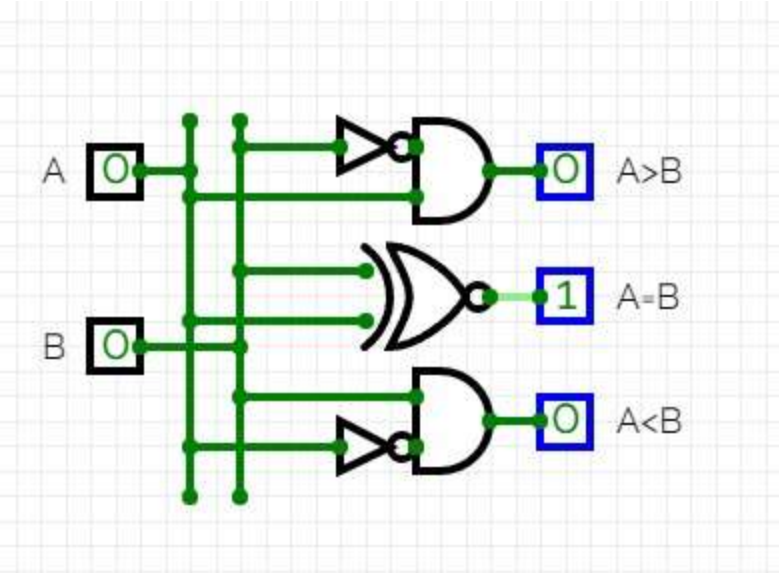
ALU_OP	Functionality
00	Addition
01	Subtraction (implemented by adding value at A with 2’s complement of value at B)

10	Shift right
11	Comparison

XIV. 8-bit-Comparator

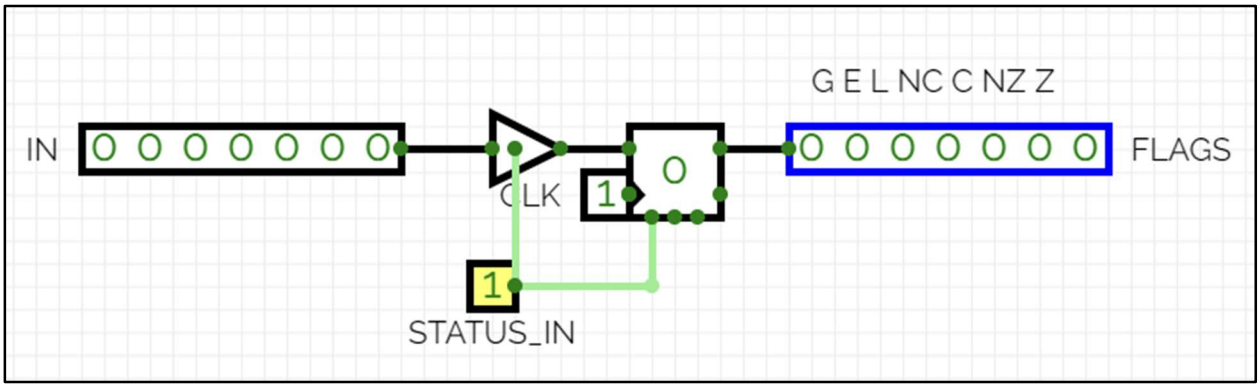


- 1-bit-Comparator



8-bit comparator is made by 2 4 – bit comparators .
4- bit comparator is made with 2 2 – bit comparators
The comparators are hence made recursively .

6. Status Register(`reg_status_7`)- Z, NZ, C, NC, L, E, G



I. The ALU transmits a 7-bit flag data through the **IN** stream. When **STATUS_IN** is set, the register stores the data from the IN stream. The output flags display the data stored in the register. The flags in the output, from LSB to MSB, are defined as follows:

Z Zero Flag

NZ Not Zero Flag

C Carry Flag

NC Not Carry Flag

L Less than Flag

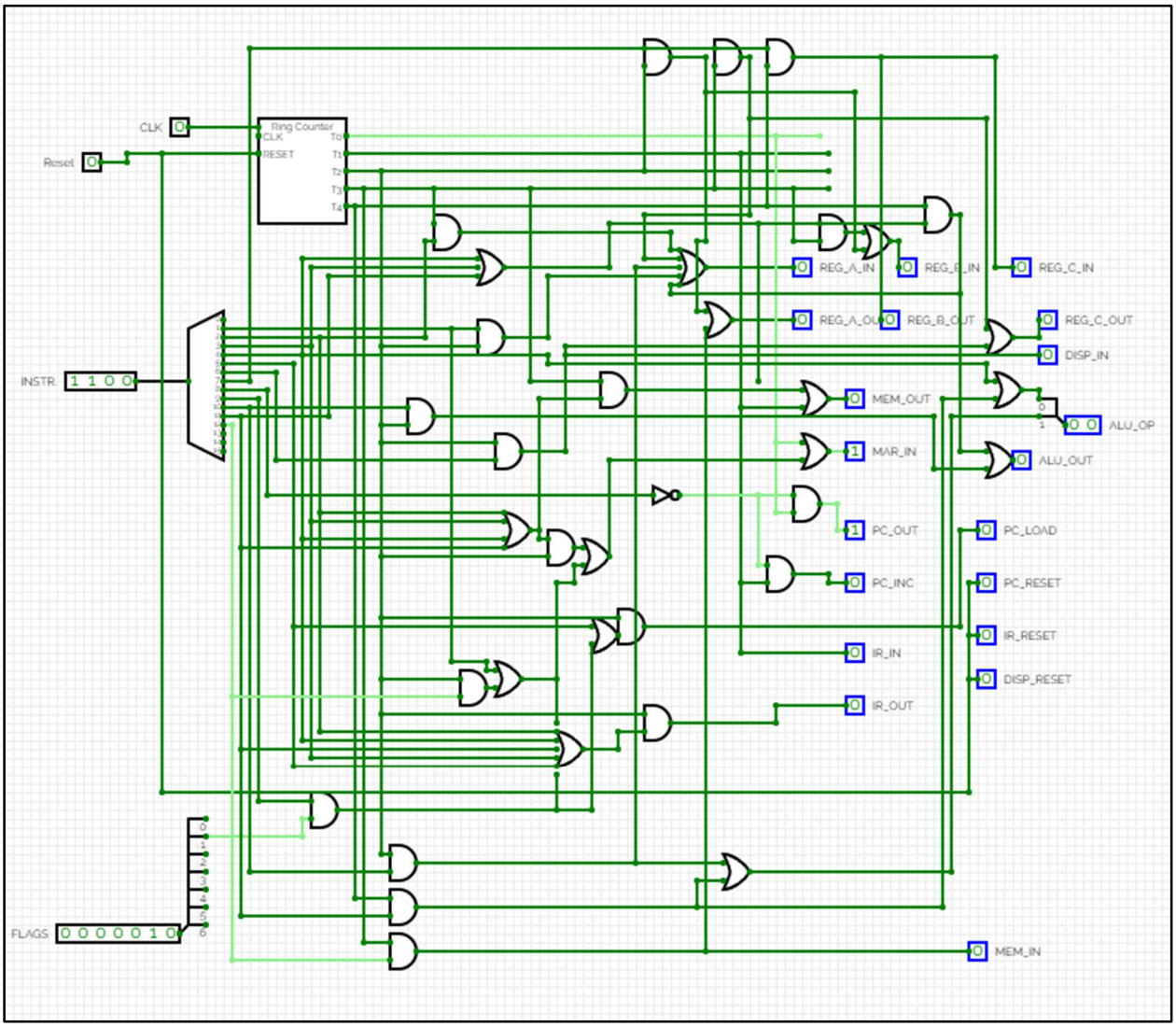
E Equal to Flag

G Greater than Flag

II. **Design** : There is a 7 - bit input stream (**IN**) which contains flags from the ALU . There is a clock input (**CLK**) and **STATUS_IN** control signal .

III. When the **STATUS_IN** is set then the status register will store the flags given as input from ALU in the 7 - bit D - flip flop at the positive clock edge .

7. CPU control module (`cpu_timing_controller`)



I. The cpu control module issues control signals to all other components of the cpu based on the instruction to be executed .

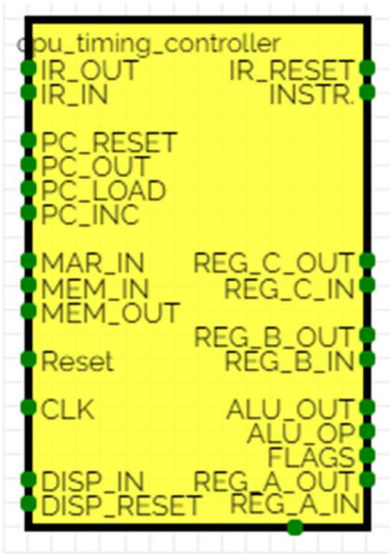
II. **Design** : The controller we designed for this cpu is a hardware controller

III. The inputs to the hardware controller are **CLK , Reset** (which resets the system) , **INSTR** (current instruction to be executed) , and **FLAGS**(from the status register)

IV. The control signals issued include :

   A. IR_OUT , IR_IN , IR_RESET
   B. PC_RESET ,PC_OUT, PC_LOAD, PC_INC
   C. MAR_IN
   D. MEM_IN, MEM_OUT

E. DISP_IN , DISP_RESET(can be used as display reset)
F. REG_C_OUT , REG_C_IN
G. REG_B_OUT, REG_B_IN
H. REG_A_OUT , REG_A_IN
I. ALU_OUT , ALU_OP ,



Instruction Set Descriptions

Instruction Set Nomenclature

Status Register(**`reg_status_7`**)

Z Zero Flag

NZ Not Zero Flag

C Carry Flag

NC Not Carry Flag

L Less than Flag

E Equal to Flag

G Greater than Flag

Registers and Operands

K: Constant data

k: Constant Address

(k): Data at address k

A: Register A (Accumulator)

B: Register B

C: Register C

Instruction Set

Assembly	Machine Code(Hex)
NOP	0x0
LDI	0x1
LDA	0x2
ADD	0x3
SUB	0x4
JMP	0x5
OUT	0x6
SWAP	0x7
HLT	0x8
JNZ	0x9
SFT	0xA
CMP	0xB

NOP

Description

This instruction performs no operation for one instruction cycle .

	Operation:		
(i)	No operation		
	Syntax	Operands	Program Counter
(i)	NOP	None	PC <- PC + 1

8 bit opcode

0000	0000

Machine Cycles

I. Total machine cycles taken by processor to complete instruction : 5

II. No operation is done . Only fetching and program counter increment takes 2 machine cycles . There is no operation done in the remaining 3 machine cycles .

Control Words/Microinstructions

NOP:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	0
	3	0
	4	0

LDI

Description

This instruction loads a 4 bit constant into the least significant 4 bits of the accumulator register A .

Operation:

(i) A <- K

	Syntax	Operands	Program Counter
(i)	LDI K	0 ≤ K ≤ 15	PC <- PC + 1

8 bit opcode

0001	KKKK

Machine Cycles

I. Minimum machine cycles / T- states required : 3
II. Machine cycles unused during instruction cycle : 2
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

LDI:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	IR_OUT, REG_A_IN
	3	0
	4	0

LDA

Description

This instruction loads the 1-byte data stored in the memory address denoted by the least significant 4 bits of the opcode into the accumulator register A.

(i) Operation:

A <- (k)

(i)	Syntax	Operands	Program Counter
	LDA K	0 ≤ k ≤ 15	PC <- PC + 1

8 bit opcode

0010	kkkk

Machine Cycles

I. Minimum machine cycles / T- states required : 4
II. Machine cycles unused during instruction cycle : 1
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

LDA:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	IR_OUT, MAR_IN
	3	MEM_OUT , REG_A_IN
	4	0

ADD

Description

This instruction performs addition of 8 bit values at registers A and Memory address k and loads the result into register A . All numbers are treated to be non-negative unsigned integers .
8 bit value at memory location k is loaded into register B .
Subsequently the result of addition of register A and register B is loaded into register A.
Original data at register B is **lost.**

Operation:

(i) B <- (k)

(ii) A <- A + B

	Syntax	Operands	Program Counter
(i)	ADD k	0 ≤ k ≤ 15	PC <- PC + 1

8 bit opcode

0011	kkkk

Status Register

[Set refers to making the value of flag to 1 , clear refers to making the flag's value 0]

Z Set if result of addition is equal to 0 , Else cleared

NZ Cleared if result of addition is equal to 0 , Else set

C Set if result of addition produces a carry , Else cleared

NC Cleared if result of addition produces a carry , Else set

L Set if 8 bit value at A was less than the 8 bit value at memory location k before any operation , else cleared

E Set if 8 bit value at A was equal to the 8 bit value at memory location k, else cleared

G Set if 8 bit value at A was greater than the 8 bit value at memory location k , else cleared

Machine Cycles

I. Minimum machine cycles / T- states required : 5
II. Machine cycles unused during instruction cycle : 0
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

ADD:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	PC_OUT ,MAR_IN
	3	MEM_OUT ,REG_B_IN
	4	ALU_OUT ,REG_A_IN , ALU_OP = 0

SUB

Description

This instruction subtracts the value at memory address specified by k from the 8 bit value at registers A and Memory address k and loads the result into register A . All numbers are treated to be non-negative unsigned integers .
8 bit value at memory location k is loaded into register B .
Subsequently the result of subtracting value at register B from value at register A is loaded into register A .
Original data at register B is **lost.**

Operation:

(i) B <- (k)

(ii) A <- A - B

	Syntax	Operands	Program Counter
(i)	SUB k	0 ≤ k ≤ 15	PC <- PC + 1

8 bit opcode

0100	kkkk

Status Register

[Set refers to making the value of flag to 1 , clear refers to making the flag's value 0]

Z Set if result of subtraction is equal to 0 , else cleared

NZ Cleared if result of subtraction is equal to 0 , else set

C Set if data at A added with two's complement of data at memory address k produces carry , else Cleared

NC Cleared if data at A added with two's complement of data at memory address k produces carry , else set

L Set if 8 bit value at A was less than the 8 bit value at memory location k before any operation ,else cleared

E Set if 8 bit value at A was equal to the 8 bit value at memory location k , else cleared

G Set if 8 bit value at A was greater than the 8 bit value at memory location k , else cleared

Machine Cycles

I. Minimum machine cycles / T- states required : 5

II. Machine cycles unused during instruction cycle : 0
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

SUB:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	PC_OUT ,MAR_IN
	3	MEM_OUT ,REG_B_IN
	4	ALU_OUT ,REG_A_IN , ALU_OP = 1

JMP

Description

Jump to an instruction at the address specified by k .

Operation:

(i) PC <- k

	Syntax	Operands	Program Counter
(i)	JMP k	0 ≤ k ≤ 15	PC <- k

8 bit opcode

0101	kkkk

Machine Cycles

I. Minimum machine cycles / T- states required : 3
II. Machine cycles unused during instruction cycle : 2
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

JMP:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	IR_OUT ,PC_LOAD
	3	0
	4	0

OUT

Description

Displays the data at Register C in preferred display mode .

As default it is a scrolling display .

Operation:

(i) I/O <- C

	Syntax	Operands	Program Counter
(i)	OUT	None	PC <- PC + 1

8 bit opcode

0110	0000

Machine Cycles

I. Minimum machine cycles / T- states required : 3
II. Machine cycles unused during instruction cycle : 2
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

OUT:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	REG_C_OUT , DISP_IN
	3	0
	4	0

SWAP

Description

Swaps the 8-bit data stored at registers A and C .
Register B is used as a temporary buffer , hence at the end of the instruction the **original** data at register B is **lost**.

	Operation:
(i)	B <- A
(ii)	A <- C
(iii)	C <- B

	Syntax	Operands	Program Counter
(i)	SWP	None	PC <- PC + 1

8 bit opcode

0111	0000

Machine Cycles

I. Minimum machine cycles / T- states required : 5
II. Machine cycles unused during instruction cycle : 0
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

SWAP:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	REG_A_OUT , REG_B_IN
	3	REG_A_IN , REG_C_OUT
	4	REG_C_IN , REG_B_OUT

HLT

Description

Halts program execution. When this instruction is loaded into the Instruction Register the processor is halted until the system is reset .

Operation:

(i) None

	Syntax	Operands	Program Counter
(i)	HLT	None	PC <- PC + 1

During the instruction cycle in which HALT instruction is being fetched into Instruction Register the Program Counter is incremented . However from the next instruction cycle no control signal is issued in any of the 5 T - states .

8 bit opcode

1000	0000

Machine Cycles

I. Minimum machine cycles / T- states required : 2
II. Machine cycles unused during instruction cycle : 3
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

HLT:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	0
	3	0
	4	0

JNZ

Description

Jumps to instruction at the address specified by k if the NZ flag is set , otherwise it continues instructions in order.

Operation:

(i) if(NZ)PC <- k
else PC <- PC + 1

	Syntax	Operands	Program Counter
(i)	JNZ	0 ≤ k ≤ 15	if(NZ)PC <- k else PC <- PC + 1

8 bit opcode

1001	0000

Machine Cycles

I. Minimum machine cycles / T- states required : 3
II. Machine cycles unused during instruction cycle : 2
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

JNZ:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	if NZ flag is set : IR_OUT , PC_LOAD else : none
	3	0
	4	0

SFT(Right)

Description

This instruction performs a bitwise right shift operation on the 1-byte data stored in the accumulator. After the shift operation is performed once the MSB of the data in the accumulator is Cleared.

Operation:

(i) A <- A>>1

	Syntax	Operands	Program Counter
(i)	SFT	None	PC <- PC + 1

8 bit opcode

1010	0000

Status Register

[Set refers to making the value of flag to 1 , clear refers to making the flag's value 0]

Z Set if result of shifting 8 - bit value at A by one bit rightward is equal to 0 , Else cleared

NZ Cleared if result of shifting 8 - bit value at A by one bit rightward is equal to 0 , Else set

C Cleared

NC Set

L Set if 8 - bit value at register A is less than the 8 - bit value at register B before shift operation ,Else cleared

E Set if 8 - bit value at register A is equal to the 8 - bit value at register B before shift operation ,Else cleared

G Set if 8 - bit value at register A is greater to the 8 - bit value at register B before shift operation ,Else cleared

Machine Cycles

I. Minimum machine cycles / T- states required : 3
II. Machine cycles unused during instruction cycle : 2
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

SFT:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	REG_A_IN , ALU_OP = 2
	3	0
	4	0

CMP

Description

This instruction compares the value in register A with the data stored at memory address k and sets the greater, less or equality flag to 1 if it is applicable.
8 bit value at memory location k is loaded into register B .
Subsequently the flags of status register are set on comparing data at register A and B.
Original data at register B is **lost.**

Operation:

(i) B <- (k)

	Syntax	Operands	Program Counter
(i)	CMP k	0 ≤ k ≤ 15	PC <- PC + 1

8 bit opcode

1011	kkkk

Status Register

[Set refers to making the value of flag to 1 , clear refers to making the flag's value 0]

Z Set if A is equal to 0 , Else cleared

NZ Cleared if A is equal to 0 , Else set

C Cleared

NC Set

L Set if 8 bit value at A is less than the 8 bit value at memory location k ,Else cleared

E Set if 8 bit value at A is equal to the 8 bit value at memory location k, else cleared

G Set if 8 bit value at A is greater than the 8 bit value at memory location k , else cleared

Machine Cycles

I. Minimum machine cycles / T- states required : 5
II. Machine cycles unused during instruction cycle : 0
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

SFT:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	IR_OUT, MAR_IN
	3	MEM_OUT, REG_B_IN
	4	ALU_OP = 3

STA

Description

This instruction stores a 8 - bit value from Accumulator into the memory module . Note that M[k] represents the memory at k address in memory .

Operation:

(i) M[k] <- A

	Syntax	Operands	Program Counter
(i)	STA k	0 ≤ k ≤ 15	PC <- PC + 1

8 bit opcode

1100	kkkk

Machine Cycles

I. Minimum machine cycles / T- states required : 4
II. Machine cycles unused during instruction cycle : 1
III. Total machine cycles taken by processor to complete instruction : 5

Control Words/Microinstructions

STA:	T - state	Control Signals Issued (Any control signal not mentioned are 0)
	0	PC_OUT ,MAR_IN
	1	MEM_OUT , IR_IN , PC_INC
	2	IR_OUT, MAR_IN
	3	MEM_IN, REG_A_OUT
	4	None

Few programs implemented using the CPU

Fill some values in address c, d, e, f of the ROM.
1) Adding 2 numbers and displaying the result in scroll display

Operation (Assembly)	Op Code
LDI 0x6	0x16
ADD 0xF	0x3F
SWAP	0x70
OUT	0x60
HLT	0x80

2) Adding and subtracting four numbers in some combination

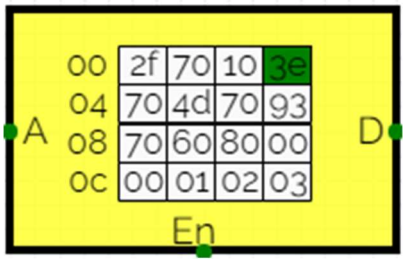
Operation (Assembly)	Op Code
LDI 0x5	0x15
ADD 0xF	0x3F
SUB 0xE	0x4E

ADD 0xD	0x3D
HLT	0x80

3) A multiplication routine using repeated addition

Set the value at address 0xD = 0x01,
Location 0xE contains a
Location 0xF contains b
The result of multiplication of a and b is displayed after program is run .

Operation (Assembly)	Op Code
LDA 0xF	0x2F
SWAP	0x70
LDI 0x0	0x10
ADD 0xE	0x3E
SWAP	0x70
SUB 0xD	0x4D
SWAP	0x70
JNZ 0x3	0x93
SWAP	0x70
OUT	0x60
HLT	0x80



Above ROM program with data is run which outputs 06 in display which is the correct product of 2 x 3 .

4) Integer division by 4 and displaying the output

Operation (Assembly)	Op Code
LDI 0x9	0x19
SFT	0xA0
SFT	0xA0
SWAP	0x70
OUT	0x60
HLT	0x80

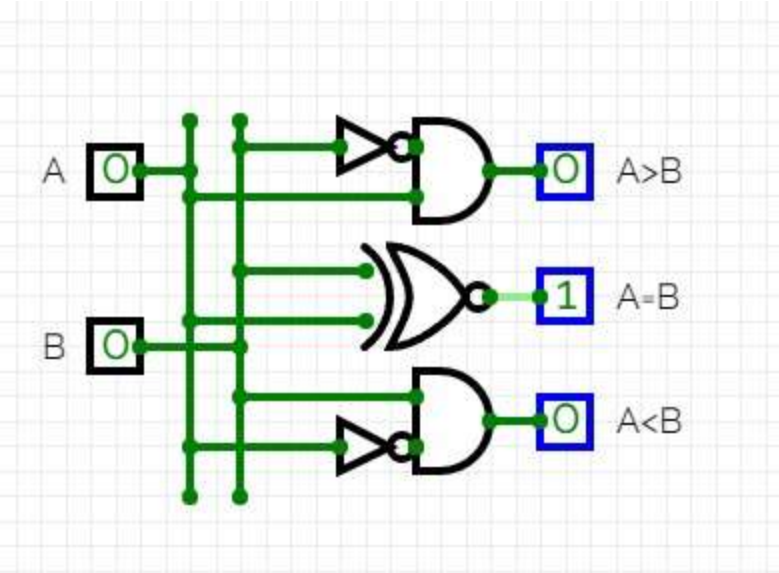
5) Outputs numbers from certain x to 1 in display .
The number x must be stored at memory location 0xF .
The number 1 must be stored at memory location 0xE .

Operation (Assembly)	Op Code
LDA 0xF	0x2F
SWAP	0x70
OUT	0x60
SWAP	0x70

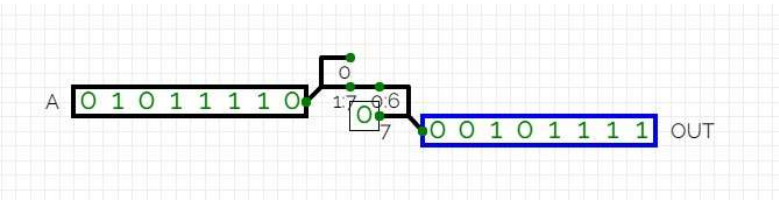
SUB 0xE	0x4E
SWAP	0X70
JNZ 0x2	0x92
HLT	0x80

Extra Credit :

1)Updated ALU to support SHIFT right functionality and used in above programs to do divide by 4 (basic circuit below).
Updated ALU to include CMP (basic circuit below).
Created SWAP instruction . Used it in programs above such as multiplication program above .



1-bit Comparator



Shift Right

2)Conditional Jump (JNZ) instruction created and used several times for programs in cpu .

4) The bits from MSB to LSB in the control word for FSM design is- PC_INC, PC_OUT, PC_LOAD, MEM_OUT, IR_IN, IR_OUT, MAR_IN, A_IN, A_OUT, B_IN, B_OUT, C_IN, C_OUT, DISP_IN, ALU_OUT, ALU_OP (2 bits).

The FSM state diagram illustrates the control word. Blocks at the same height correspond to the same T-state. For an edge with no information written on it, everything within that block transitions into the next block through that edge.