

COMPUTER ARCHITECTURE (ECT-312)

PROCESSOR PROJECT - EDITH

Submitted to:

Mr. Rakesh Bairathi

Submitted by:

Himanshu Omar	(2020UEC1341)
Gungun Singhal	(2020UEC1372)
Ashutosh Mishra	(2020UEC2016)
Khanra Akash Ananta	(2020UEC1901)
Guntuku Abhilash	(2020UEC2006)



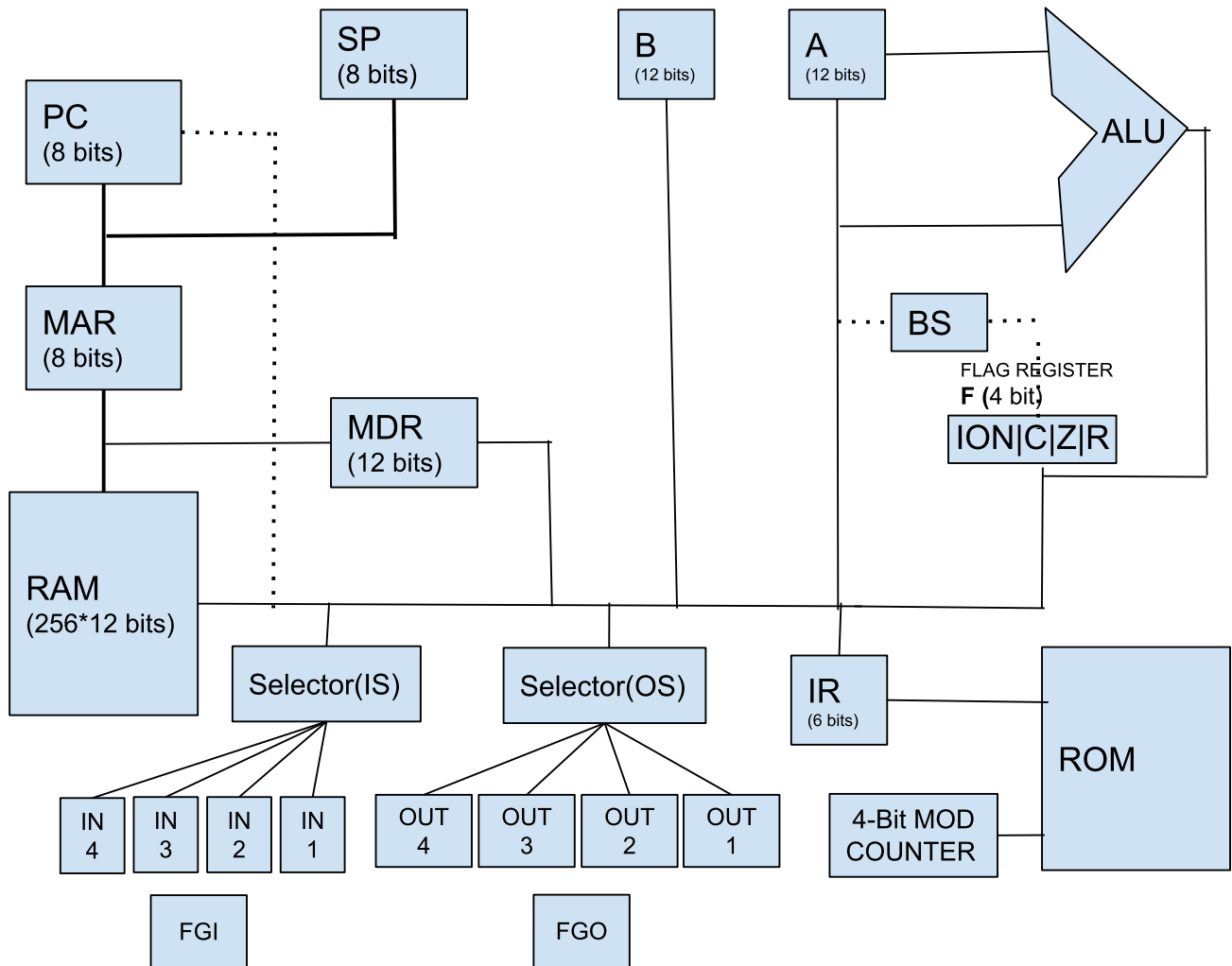
Malaviya National Institute of Technology

ELECTRONICS AND COMMUNICATION DEPARTMENT (2022-2023)

Table of Contents

Sr no.	Contents	Page no.
1.	Processor Architecture Layout	3
2.	Components Description	4
3.	Unique Features	7
4.	Instruction Set	8-9
	i.) MRI	8
	ii.) Non-MRI	9
5.	Micro-Instructions	10-21
	i.) MRI	10
	ii.) Non-MRI	15
6.	Sample Outputs	22
7.	Interrupt Handling	41
8.	Hardware Implementation	44
9.	Conclusion	46

Processor Architecture Layout



REGISTER DESCRIPTION

1. MEMORY DATA REGISTER (12-Bit) :

Instructions from RAM are fetched and transferred into this Register for further decoding and then sent to the IR register.

Format:

➤ For MRI Instruction:

1-Bit	3-Bit	8-Bit
I (Direct or Indirect)	MRI Opcode	Address

➤ For non-MRI Instruction:

1-Bit	3-Bit	4-Bit	4-Bit
I (Register or I/O based)	111	Non-MRI Opcode	Data/Don't care

2. FLAG REGISTER (4-Bit) :

Format:

4-Bit			
ION	C	Z	R

- **ION flag:-** ION is reset means processor will be handling interrupt, ION is set means processor will not be handling any interrupt.
- **Carry flag:-** C is set and reset on every instruction based on Arithmetic and Logical Operations on Accumulator.
- **Zero flag:-** Z is set and reset on instruction based on Arithmetic and Logical Operations on Accumulator when it becomes equal to zero.
- **R flag:-** R is set when an interrupt has appeared and to be serviced, R is reset when the processor has finished handling that interrupt.

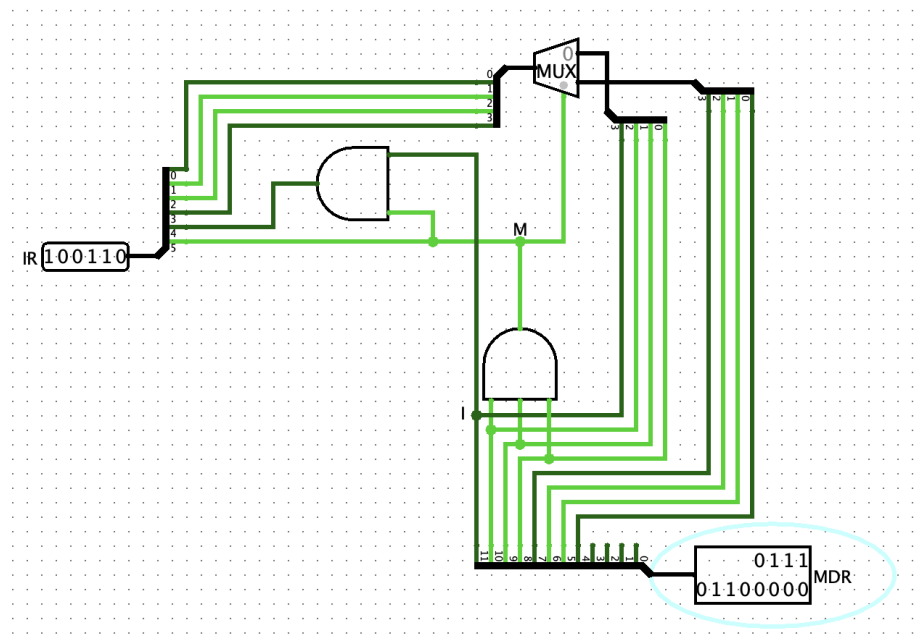
3. INSTRUCTION REGISTER (6-Bit) :

Format:

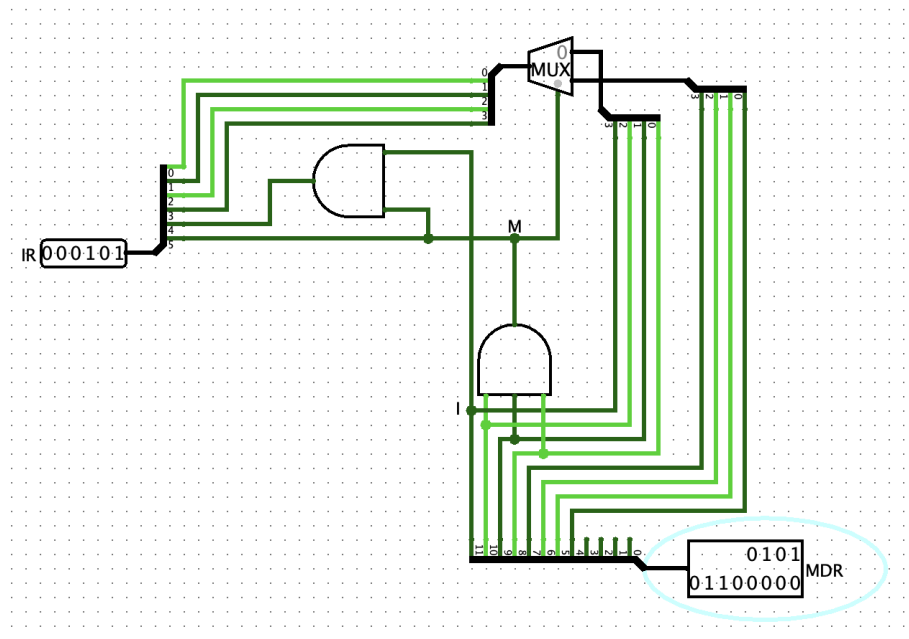
1-Bit	1-Bit	4-Bit
M (MRI/Non-MRI)	I/O (set for I/O operation)	Opcode(based on M)

- M=0 for MRI and M=1 for non-MRI
- I/O=0 for MRI and non-MRI register based instructions and I/O=MDR(11) for input-output based non-MRI instructions.
- For MRI, Opcode will be MDR(11-8) and for non-MRI, Opcode will be MDR(7-4)

Circuit Diagram:



Working Hardware : Decoding of non-MRI Instruction.



Working Hardware : Decoding of MRI Instruction

Most significant bit in IR is the “MODE” bit which will be set when there is a non-MRI instruction, i.e. when MDR(10-8) is ‘111’.
 Next to MSB there is an I/O bit, which will be set for I/O operations only.

If MODE bit is set then $IR(0-3) \leftarrow MDR(4-7)$
 else $IR(0-3) \leftarrow MDR(8-11)$

UNIQUE FEATURES OF EDITH PROCESSOR

1. Interrupt Handling:

Pressing a key “i” in between the execution of any operation will generate an interrupt.

`_kbhit()` function is implemented to read from the console. It checks the console for a recent keystroke. If the function returns a non-zero value, a keystroke is waiting in the buffer. The program can then call `_getch()` or `_getche()` to read the keystroke. By default, this function's global state is scoped to the application.

More detail in Interrupt handling section.

2. Implementation of Barrel Shifter:

It is a specialised digital electronic circuit with the purpose of shifting an entire data word by a specified number of bits by only using combinational logic, with no sequential logic used. The simplest way of achieving this is by using a series of multiplexers where one output is connected to the input of the next multiplexer in the chain, in a specific manner that depends on the amount of shift specified.

3. Multiple I/O ports:

EDITH has 4 input ports and 4 output ports. Each port can be selected only once at a time. Selection of ports is done by IS(Input Selector) and IO(Output Selector) respectively based on the last 4 bits(decoded bits) of instruction(Opcode) i.e. MDR(3-0).

4. Immediate Data Instruction:

In our Instruction Set, we kept Immediate Instruction in Non MRI, as we are not referring to memory(RAM) for Data. We give that data from Opcode itself. The last 4 bits of instruction(Opcode) i.e. MDR(3-0) will be immediately transferred to the target register.

Instruction Set

1. MRI Instruction

Format of 12-bit MRI Instruction:

I (1-Bit) (Direct or Indirect)	3-Bit MRI Opcode	8-Bit Address
-----------------------------------	------------------	---------------

Instruction Table:

MRI Instruction	Opcode	
	Direct (I=0)	Indirect (I=1)
LDA XX	0XX	8XX
STA XX	1XX	9XX
ADM XX	2XX	AXX
CALL XX	3XX	BXX
JMP XX	4XX	CXX
JC XX	5XX	DXX
JZ XX	6XX	EXX

'XX' - 8-Bit Address

2. non-MRI Instruction

Format of 12-bit non-MRI Instruction:

I (1-Bit) (Register or I/O based)	111	4-Bit Non-MRI Opcode	4-Bit Data/Don't care
--------------------------------------	-----	----------------------	-----------------------

Instruction Table:

Type	Non-MRI Instruction	Opcode
Arithmetic and Logical	ADD B	70_
	SUB B	71_
	CLA	72_
	CMA	73_
Register based Data Transfer	MOV A,B	74_
	MOV B,A	75_
Stack related	PUSH A	76_
	POP	77_
	RET	78_
* Immediate Data Transfer	MVI A,data	79X
	MVI F,data	7AX
Shift and Rotate related	LSC A,data	7BX
	RSC A,data	7CX
	RRX A,data	7DX
	ASR A,data	7EX
Halt	HLT	7F_
** I/O related (I/O=1)	IN S	F0X
	OUT S	F1X
	SKI S	F2X
	SKO S	F3X

'X' - 4-Bit Data

'_' - can be any value from 0 to F

** 'S' - 4-Bit Data to Select I/O Ports only take values(1,2,4,8)

MICRO INSTRUCTIONS

For MRI Instructions

LDA

Instruction to load register A with the value at the given address(direct)

Direct : instruction_format (0XX)

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

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

T2 : MDR(E) ,IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(8-11)

T3 : Idle

T4 : Idle

T5 : A(L) , RAM(E), Reset // A <- M[MAR]

Indirect: instruction_format (8XX)

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

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

T2 : MDR(E) , IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(11-8)

T3 : MDR(L) , RAM(E) // MDR <- M[MAR]

T4 : MAR(L,E), MDR(E) // MAR <- MDR(7-0)

T5 : A(L) , RAM(E), Reset // A <- M[MAR]

STA

Instruction to store the value from register A to the given memory location.

Direct : instruction_format (1XX)

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

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

T2 : MDR(E) ,IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(8-11)

T3 : Idle

T4: Idle
T5: RAM(L) , A(E) , Reset // M[MAR] <- A

Indirect: instruction_format (**9XX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(11-8)
T3: MDR(L) , RAM(E) // MDR <- M[MAR]
T4: MAR(L,E), MDR(E) // MAR <- MDR(7-0)
T5: RAM(L) , A(E) , Reset // M[MAR] <- A

ADM

Instruction to add AC to memory operand.

Direct : instruction_format (**2XX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) , MAR(L) // MAR <- MDR(7-0), IR <- MDR(8-11)
T3: Idle
T4: Idle
T5: MDR(L) , RAM(E) // MDR <- M[MAR]
T6: A(E,L) , MDR(E) , ALU(000) , Reset // A <- A + MDR

Indirect: instruction_format (**AXX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(11-8)
T3: MDR(L) , RAM(E) // MDR <- M[MAR]
T4: MAR(L,E),MDR(E) // MAR <- MDR(7-0)
T5: MDR(L) , RAM(E) // MDR <- M[MAR]
T6: A(E,L) , MDR(E) , ALU(000) , Reset // A <- A + MDR

CALL

The CALL microoperation stores the return address in the stack and loads the PC with the given address.

Direct : instruction_format (**3XX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E), IR(L), MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(11-8)
T3: Idle
T4: Idle
T5: SP(D)
T6: SP(E), MAR(E,L) // MAR <- SP
T7: RAM(L), PC(E) // M[MAR] <- PC
T8: PC(L), MDR(E), Reset // PC <- MDR(7-0)

Indirect: instruction_format (**BXX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I), RAM(E), MDR(L)
T2: MDR(E), IR(L), MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(11-8)
T3: MDR(L), RAM(E) // MDR <- M[MAR]
T4: Idle
T5: SP(D)
T6: SP(E), MAR(E,L) // MAR <- SP
T7: RAM(L), PC(E) // M[MAR] <- PC
T8: PC(L), MDR(E) // PC <- MDR(7-0)

JMP

The program sequence is transferred to the memory location specified by the particular level given in the operand.

Direct : instruction_format (**4XX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L), MAR(L,E) // IR <- MDR(8-11)

T3: Idle
T4: Idle
T5: PC(L) , MDR(E), Reset // PC <- MDR(7-0)

Indirect: instruction_format (**CXX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) , IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(11-8)
T3: MDR(L) , RAM(E) // MDR <- M[MAR]
T4: Idle
T5: PC(L) , MDR(E), Reset // PC <- MDR(7-0)

JC

The program sequence is transferred to a particular level or a 12-bit address if C=1 (or carry is 1)

Direct : instruction_format (**5XX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L), MAR(L,E) // IR <- MDR(8-11)
T3: Idle
T4: Idle
T5: PC(L) , MDR(E), Reset // if C=1, PC <- MDR(7-0)
else Reset

Indirect: instruction_format (**DXX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) , IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <-MDR(11-8)
T3: MDR(L) , RAM(E) // MDR <- M[MAR]
T4: Idle
T5: PC(L) , MDR(E), Reset // if C=1, PC <- MDR(7-0)
else Reset

JZ

The program sequence is transferred to a particular level or a 12-bit address if Z=1 (or zero flag is 1)

Direct : instruction_format (**6XX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2 : MDR(E) , IR(L) // IR <- MDR(8-11)
T3 : Idle
T4 : Idle
T5 : PC(L) , MDR(E), Reset // if Z=1, PC <- MDR(7-0)
else Reset

Indirect: instruction_format (**EXX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2 : MDR(E) , IR(L) , MAR(L,E) // MAR <- MDR(7-0), IR <- MDR(11-8)
T3 : MDR(L) , RAM(E) // MDR <- M[MAR]
T4 : Idle
T5 : PC(L) , MDR(E), Reset // if Z=1, PC <- MDR(7-0)
else Reset

For non-MRI Instructions

ADD B : instruction_format (70_)

Instruction to add values present in RAM/Registers.

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

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

T2: MDR(E) ,IR(L) // IR <- MDR(4-7)

T3: Idle

T4: Idle

T5: A(E,L) , B(E) , ALU(000) , Reset

SUB B : instruction_format (71_)

Instruction to subtract values present in RAM/Registers.

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

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

T2: MDR(E) ,IR(L) // IR <- MDR(4-7)

T3: Idle

T4: Idle

T5: A(E,L) , B(E) , ALU(001) , Reset

CLA : instruction_format (72_)

This instruction specifies to clear the accumulator.

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

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

T2: MDR(E) ,IR(L) // IR <- MDR(4-7)

T3: Idle

T4: Idle

T5: A(E,L) , ALU(002) , Reset

CMA : instruction_format (73_)

This instruction specifies to complement the value present in the accumulator.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <- MDR(4-7)
T3: Idle
T4: Idle
T5: A(E,L) , ALU(003) , Reset

MOV A,B : instruction_format (74_)

This instruction specifies to move the data present in register B to register A.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <- MDR(4-7)
T3: Idle
T4: Idle
T5: B(E), A(L) , Reset // A <- B

MOV B,A : instruction_format (75_)

This instruction specifies to move the data present in register A to register B.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <- MDR(4-7)
T3: Idle
T4: Idle
T5: B(L), A(E) , Reset // B <- A

PUSH : instruction_format (76_)

This instruction is used to push the data from A to the address pointed by SP in RAM.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
T3: Idle
T4: Idle
T5: SP(D)
T6: SP(E) , MAR(E,L) // MAR <- SP
T7: RAM(L) , A(E) , Reset // M[MAR] <- A

POP : instruction_format (77_)

This instruction is used to pop the data from the address pointed by SP in RAM.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
T3: Idle
T4: Idle
T5: SP(E) , MAR(E,L) // MAR <- SP
T6: RAM(E) , A(L) // A <- M[MAR]
T7: SP(I) , Reset

RET : instruction_format (78_)

Retrieves PC from the address pointed by SP in RAM.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
T3: Idle
T4: Idle
T5: SP(E) , MAR(E,L) // MAR <- SP
T6: RAM(E) , PC(L) // PC <- M[MAR]
T7: SP(I) , Reset

MVI A , data : instruction_format (**79X**)

This instruction specifies to move the data immediately in A.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
T3: Idle
T4: Idle
T5: MDR(E), A(L) , Reset // A <- MDR(0-3)

MVI F , data : instruction_format (**7AX**)

This instruction specifies to move the data immediately in F.

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
T3: Idle
T4: Idle
T5: MDR(E), F(L) , Reset // F <- MDR(0-3)

LSC A , data : instruction_format (**7BX**)

T0 : PC(E) , MAR(E,L)
T1 : PC(I) , RAM(E) , MDR(L)
T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
T3: Idle
T4: Idle
T5: MDR(E), BS(L) // BS <- MDR(0-3)
T6: A(E,L), BS(E), Reset

RSC A , data : instruction_format (**7CX**)

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: MDR(E), BS(L) // BS <- MDR(0-3)

T6: A(E,L), BS(E), Reset

RRX A , data : instruction_format (**7DX**)

Input carry, A and tmp value which has one time to rotate to the barrel shifter and note the answer in the accumulator

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: MDR(E), BS(L) // BS <- MDR(0-3)

T6: A(E,L), BS(E), Reset

ASR A , data : instruction_format (**7EX**)

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: MDR(E), BS(L) // BS <- MDR(0-3)

T6: A(E,L), BS(E), Reset

HLT : instruction_format (**7F_**)

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: disable SG , Reset // SG - sequence generator

IN S : instruction_format (**F0X**)

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: MDR(E), IS(L) // IS <- MDR(0-3)

T6: A(L), IN(E), FGI(0) , Reset // A <- IN, FGI <- 0,
(IN Port selected by IS)

OUT S : instruction_format (**F1X**)

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: MDR(E), OS(L) // OS <- MDR(0-3)

T6: A(E), OUT(L), FGO(0), Reset // OUT <- A, FGO <- 0,
(OUT Port selected by OS)

SKI S : instruction_format (**F2X**)

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: MDR(E), IS(L) // IS <- MDR(0-3)

T6: PC(I), Reset // if FGI=1,PC<-PC+1, (FGI is selected by IS)
else Reset

SKO S : instruction_format (**F3X**)

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

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

T2: MDR(E) ,IR(L) // IR <-MDR(4-7)

T3: Idle

T4: Idle

T5: MDR(E), OS(L) // OS <- MDR(0-3)

T6: PC(I), Reset // if FGO=1,PC<-PC+1,(FGO is selected by OS)
else Reset

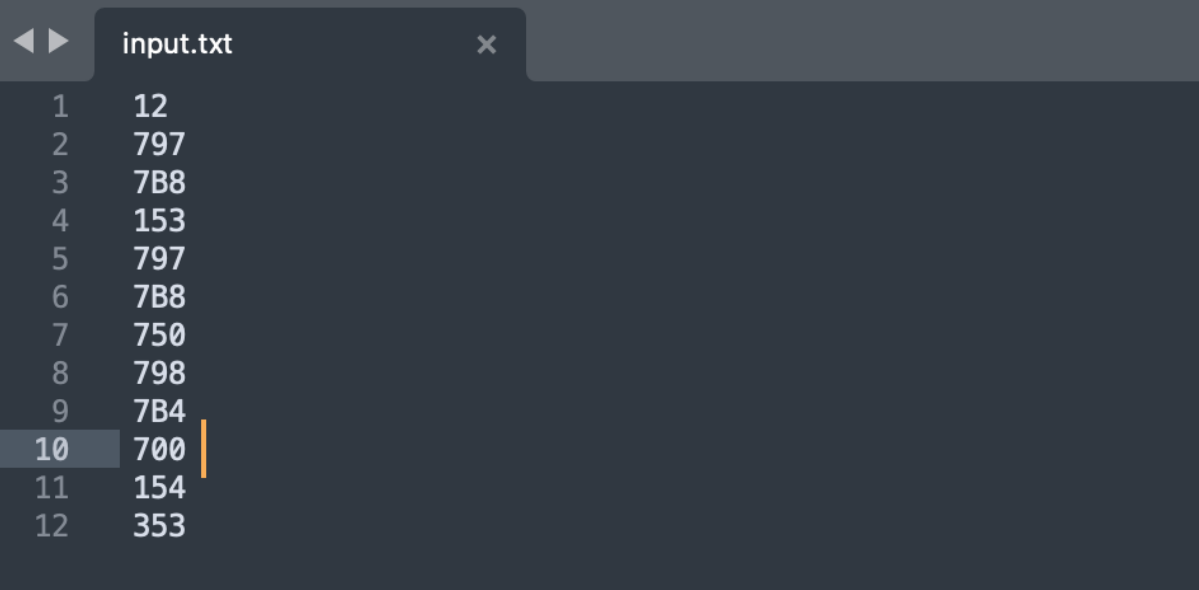
Sample Outputs

1. CALL 0x53

Input File:-

Number of Instructions :-

```
MVI A,7
LSC 8
STA 0X53
MVI A,7
LSC 8
MOV B,A
MVI A,8
LSC 4
ADD B
STA 0X54
CALL 0X53
HLT
```



The screenshot shows a text editor window titled 'input.txt'. The editor contains a list of 12 instructions, each with a line number and a hexadecimal value. The 10th line is highlighted with a blue background and an orange cursor. The instructions are as follows:

Line	Instruction
1	12
2	797
3	7B8
4	153
5	797
6	7B8
7	750
8	798
9	7B4
10	700
11	154
12	353

Output File:-

```
output.txt x
1  ~Fetch:
2  T0 : PC(E) , MAR(E,L)
3  T1 : PC(I) , RAM(E) , MDR(L)
4
5  PC=01 IR=00 MDR=797 MAR=00 SP=FF A=700 B=000
6  (ION=0 C=0 Z=0 R=0) FLG=0
7  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
8  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
9
10 ~Decode:
11 This is non MRI Instruction
12 T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
13
14 PC=01 IR=29 MDR=797 MAR=00 SP=FF A=700 B=000
15 (ION=0 C=0 Z=0 R=0) FLG=0
16 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
17 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
18
19 ~Execution:
20
21 * * * * * MVI A,7 * * * * *
22
23 T3: Idle
24 T4: Idle
25 T5: MDR(E) ,A(L), RESET
26
27 PC=01 IR=29 MDR=797 MAR=00 SP=FF A=007 B=000
28 (ION=0 C=0 Z=0 R=0) FLG=0
29 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
30 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
31
```

```

33  --- NEXT INSTRUCTION ---
34
35  ~Fetch:
36  T0 : PC(E) , MAR(E,L)
37  T1 : PC(I) , RAM(E) , MDR(L)
38
39  PC=02 IR=29 MDR=7B8 MAR=01 SP=FF A=007 B=000
40  (ION=0 C=0 Z=0 R=0) FLG=0
41  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
42  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
43
44  ~Decode:
45  This is non MRI Instruction
46  T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
47
48  PC=02 IR=2B MDR=7B8 MAR=01 SP=FF A=007 B=000
49  (ION=0 C=0 Z=0 R=0) FLG=0
50  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
51  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
52
53  ~Execution:
54
55  * * * * * LSC A,8 * * * * *
56  T6:  A(E,L), BS(E), Reset
57
58  PC=02 IR=2B MDR=7B8 MAR=01 SP=FF A=700 B=000
59  (ION=0 C=0 Z=0 R=0) FLG=0
60  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
61  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
62

```



```

64  --- NEXT INSTRUCTION ---
65
66  ~Fetch:
67  T0 : PC(E) , MAR(E,L)
68  T1 : PC(I) , RAM(E) , MDR(L)
69
70  PC=03 IR=2B MDR=153 MAR=02 SP=FF A=700 B=000
71  (ION=0 C=0 Z=0 R=0) FLG=0
72  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
73  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
74
75  ~Decode:
76  This is MRI Instruction
77  T2: MDR(E) ,IR(L) , MAR(L,E)          // MAR <- MDR(7-0), IR <-MDR(8-11)
78
79  PC=03 IR=01 MDR=153 MAR=53 SP=FF A=700 B=000
80  (ION=0 C=0 Z=0 R=0) FLG=0
81  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
82  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
83
84  ~Execution:
85
86  * * * * * Direct STA 53 * * * * *
87
88  T3: Idle
89  T4: Idle
90  T5: RAM(L) , A(E), Reset
91
92  Accumulaor Content--700 is loaded in Memory location 0X53
93
94  PC=03 IR=01 MDR=153 MAR=53 SP=FF A=700 B=000
95  (ION=0 C=0 Z=0 R=0) FLG=0
96  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
97  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
98

```

```

100  --- NEXT INSTRUCTION ---
101
102  ~Fetch:
103  T0 : PC(E) , MAR(E,L)
104  T1 : PC(I) , RAM(E) , MDR(L)
105
106  PC=04 IR=01 MDR=797 MAR=03 SP=FF A=700 B=000
107  (ION=0 C=0 Z=0 R=0) FLG=0
108  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
109  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
110
111  ~Decode:
112  This is non MRI Instruction
113  T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
114
115  PC=04 IR=29 MDR=797 MAR=03 SP=FF A=700 B=000
116  (ION=0 C=0 Z=0 R=0) FLG=0
117  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
118  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
119
120  ~Execution:
121
122  * * * * * MVI A,7 * * * * *
123
124  T3: Idle
125  T4: Idle
126  T5: MDR(E) ,A(L), RESET
127
128  PC=04 IR=29 MDR=797 MAR=03 SP=FF A=007 B=000
129  (ION=0 C=0 Z=0 R=0) FLG=0
130  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
131  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
132

```

```

134 ---- NEXT INSTRUCTION ----
135
136 ~Fetch:
137 T0 : PC(E) , MAR(E,L)
138 T1 : PC(I) , RAM(E) , MDR(L)
139
140 PC=05 IR=29 MDR=7B8 MAR=04 SP=FF A=007 B=000
141 (ION=0 C=0 Z=0 R=0) FLG=0
142 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
143 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
144
145 ~Decode:
146 This is non MRI Instruction
147 T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
148
149 PC=05 IR=2B MDR=7B8 MAR=04 SP=FF A=007 B=000
150 (ION=0 C=0 Z=0 R=0) FLG=0
151 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
152 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
153
154 ~Execution:
155
156 * * * * * LSC A,8 * * * * *
157 T6: A(E,L), BS(E), Reset
158
159 PC=05 IR=2B MDR=7B8 MAR=04 SP=FF A=700 B=000
160 (ION=0 C=0 Z=0 R=0) FLG=0
161 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
162 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
163
164

```

```

165 ---- NEXT INSTRUCTION ----
166
167 ~Fetch:
168 T0 : PC(E) , MAR(E,L)
169 T1 : PC(I) , RAM(E) , MDR(L)
170
171 PC=06 IR=2B MDR=750 MAR=05 SP=FF A=700 B=000
172 (ION=0 C=0 Z=0 R=0) FLG=0
173 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
174 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
175
176 ~Decode:
177 This is non MRI Instruction
178 T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
179
180 PC=06 IR=25 MDR=750 MAR=05 SP=FF A=700 B=000
181 (ION=0 C=0 Z=0 R=0) FLG=0
182 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
183 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
184
185 ~Execution:
186
187 * * * * * * * * MOV B,A * * * * * * * *
188
189 T3: Idle
190 T4: Idle
191 T5:  B(L) , A(E) , Reset
192
193 PC=06 IR=25 MDR=750 MAR=05 SP=FF A=700 B=700
194 (ION=0 C=0 Z=0 R=0) FLG=0
195 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
196 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
197
198

```

```

199  --- NEXT INSTRUCTION ---
200
201  ~Fetch:
202  T0 : PC(E) , MAR(E,L)
203  T1 : PC(I) , RAM(E) , MDR(L)
204
205  PC=07 IR=25 MDR=798 MAR=06 SP=FF A=700 B=700
206  (ION=0 C=0 Z=0 R=0) FLG=0
207  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
208  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
209
210  ~Decode:
211  This is non MRI Instruction
212  T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
213
214  PC=07 IR=29 MDR=798 MAR=06 SP=FF A=700 B=700
215  (ION=0 C=0 Z=0 R=0) FLG=0
216  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
217  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
218
219  ~Execution:
220
221  * * * * * MVI A,8 * * * * *
222
223  T3: Idle
224  T4: Idle
225  T5: MDR(E) ,A(L), RESET
226
227  PC=07 IR=29 MDR=798 MAR=06 SP=FF A=008 B=700
228  (ION=0 C=0 Z=0 R=0) FLG=0
229  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
230  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
231
232

```

```

233 --- NEXT INSTRUCTION ---
234
235 ~Fetch:
236 T0 : PC(E) , MAR(E,L)
237 T1 : PC(I) , RAM(E) , MDR(L)
238
239 PC=08 IR=29 MDR=7B4 MAR=07 SP=FF A=008 B=700
240 (ION=0 C=0 Z=0 R=0) FLG=0
241 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
242 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
243
244 ~Decode:
245 This is non MRI Instruction
246 T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
247
248 PC=08 IR=2B MDR=7B4 MAR=07 SP=FF A=008 B=700
249 (ION=0 C=0 Z=0 R=0) FLG=0
250 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
251 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
252
253 ~Execution:
254
255 * * * * * LSC A,4 * * * * *
256 T6: A(E,L), BS(E), Reset
257
258 PC=08 IR=2B MDR=7B4 MAR=07 SP=FF A=080 B=700
259 (ION=0 C=0 Z=0 R=0) FLG=0
260 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
261 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
262
263

```

```

263
264 --- NEXT INSTRUCTION ---
265
266 ~Fetch:
267 T0 : PC(E) , MAR(E,L)
268 T1 : PC(I) , RAM(E) , MDR(L)
269
270 PC=09 IR=2B MDR=700 MAR=08 SP=FF A=080 B=700
271 (ION=0 C=0 Z=0 R=0) FLG=0
272 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
273 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
274
275 ~Decode:
276 This is non MRI Instruction
277 T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
278
279 PC=09 IR=20 MDR=700 MAR=08 SP=FF A=080 B=700
280 (ION=0 C=0 Z=0 R=0) FLG=0
281 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
282 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
283
284 ~Execution:
285
286 * * * * * ADD B * * * * *
287
288 T3: Idle
289 T4: Idle
290 T5: A(E,L) , B(E) , ALU(000) , Reset
291
292 PC=09 IR=20 MDR=700 MAR=08 SP=FF A=780 B=700
293 (ION=0 C=0 Z=0 R=0) FLG=0
294 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
295 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
296
297

```

```

297
298 ---- NEXT INSTRUCTION ----
299
300 ~Fetch:
301 T0 : PC(E) , MAR(E,L)
302 T1 : PC(I) , RAM(E) , MDR(L)
303
304 PC=0A IR=20 MDR=154 MAR=09 SP=FF A=780 B=700
305 (ION=0 C=0 Z=0 R=0) FLG=0
306 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
307 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
308
309 ~Decode:
310 This is MRI Instruction
311 T2: MDR(E) ,IR(L) , MAR(L,E)          // MAR <- MDR(7-0), IR <-MDR(8-11)
312
313 PC=0A IR=01 MDR=154 MAR=54 SP=FF A=780 B=700
314 (ION=0 C=0 Z=0 R=0) FLG=0
315 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
316 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
317
318 ~Execution:
319
320 * * * * * Direct STA 54 * * * * *
321
322 T3: Idle
323 T4: Idle
324 T5: RAM(L) , A(E), Reset
325
326 Accumulaor Content--780 is loaded in Memory location 0X54
327
328 PC=0A IR=01 MDR=154 MAR=54 SP=FF A=780 B=700
329 (ION=0 C=0 Z=0 R=0) FLG=0
330 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
331 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
332

```



```

334 ---- NEXT INSTRUCTION ----
335
336 ~Fetch:
337 T0 : PC(E) , MAR(E,L)
338 T1 : PC(I) , RAM(E) , MDR(L)
339
340 PC=0B IR=01 MDR=353 MAR=0A SP=FF A=780 B=700
341 (ION=0 C=0 Z=0 R=0) FLG=0
342 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
343 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
344
345 ~Decode:
346 This is MRI Instruction
347 T2: MDR(E) ,IR(L) , MAR(L,E)          // MAR <- MDR(7-0), IR <-MDR(8-11)
348
349 PC=0B IR=03 MDR=353 MAR=53 SP=FF A=780 B=700
350 (ION=0 C=0 Z=0 R=0) FLG=0
351 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
352 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
353
354 ~Execution:
355
356 * * * * * Direct CALL 53 * * * * *
357
358 T3: Idle
359 T4: Idle
360 T5: SP(D)
361 T6: SP(E), MAR(E,L)
362 T7: RAM(L), PC(E)
363 T8: PC(L),MDR(E), RESET
364
365 PC=53 IR=03 MDR=353 MAR=FE SP=FE A=780 B=700
366 (ION=0 C=0 Z=0 R=0) FLG=0
367 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
368 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
369
370

```

```

370
371 ---- NEXT INSTRUCTION ----
372
373 ~Fetch:
374 T0 : PC(E) , MAR(E,L)
375 T1 : PC(I) , RAM(E) , MDR(L)
376
377 PC=54 IR=03 MDR=700 MAR=53 SP=FE A=780 B=700
378 (ION=0 C=0 Z=0 R=0) FLG=0
379 IN1=005 IN2=000 IN3=000 IN4=000 --- FGI=0110
380 OUT1=000 OUT2=000 OUT3=000 OUT4=000 --- FGO=0000
381
382 ~Decode:
383 This is non MRI Instruction
384 T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
385
386 PC=54 IR=20 MDR=700 MAR=53 SP=FE A=780 B=700
387 (ION=0 C=0 Z=0 R=0) FLG=0
388 IN1=005 IN2=000 IN3=000 IN4=000 --- FGI=0110
389 OUT1=000 OUT2=000 OUT3=000 OUT4=000 --- FGO=0000
390
391 ~Execution:
392
393 * * * * * ADD B * * * * *
394
395 T3: Idle
396 T4: Idle
397 T5: A(E,L) , B(E) , ALU(000) , Reset
398
399 PC=54 IR=20 MDR=700 MAR=53 SP=FE A=E80 B=700
400 (ION=0 C=0 Z=0 R=0) FLG=0
401 IN1=005 IN2=000 IN3=000 IN4=000 --- FGI=0110
402 OUT1=000 OUT2=000 OUT3=000 OUT4=000 --- FGO=0000
403
404

```

```

405  --- NEXT INSTRUCTION ---
406
407  ~Fetch:
408  T0 : PC(E) , MAR(E,L)
409  T1 : PC(I) , RAM(E) , MDR(L)
410
411  PC=55 IR=20 MDR=780 MAR=54 SP=FE A=E80 B=700
412  (ION=0 C=0 Z=0 R=0) FLG=0
413  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
414  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
415
416  ~Decode:
417  This is non MRI Instruction
418  T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
419
420  PC=55 IR=28 MDR=780 MAR=54 SP=FE A=E80 B=700
421  (ION=0 C=0 Z=0 R=0) FLG=0
422  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
423  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
424
425  ~Execution:
426
427  * * * * * RET * * * * *
428
429  T3: Idle
430  T4: Idle
431  T5: SP(E) ,MAR(E,L)
432  T6: RAM(E) , PC(L)
433  T7: SP(I) ,RESET
434
435  PC=0B IR=28 MDR=780 MAR=FE SP=FF A=E80 B=700
436  (ION=0 C=0 Z=0 R=0) FLG=0
437  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
438  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
439

```

```

441  --- NEXT INSTRUCTION ---
442
443  ~Fetch:
444  T0 : PC(E) , MAR(E,L)
445  T1 : PC(I) , RAM(E) , MDR(L)
446
447  PC=0C IR=28 MDR=7FF MAR=0B SP=FF A=E80 B=700
448  (ION=0 C=0 Z=0 R=0) FLG=0
449  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
450  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
451
452  ~Decode:
453  This is non MRI Instruction
454  T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
455
456  PC=0C IR=2F MDR=7FF MAR=0B SP=FF A=E80 B=700
457  (ION=0 C=0 Z=0 R=0) FLG=0
458  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
459  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000
460
461  * * * * * HALT has occured * * * * *
462
463  PC=0C IR=2F MDR=7FF MAR=0B SP=FF A=E80 B=700
464  (ION=0 C=0 Z=0 R=0) FLG=0
465  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
466  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FG0=0000

```

2. ASR 3

Input File:-

Number of Instructions :- 2

ASR 3

HLT

```

input.txt
1 2
2 7E3
3 7FF

```

Output File:-

```

1  ~Fetch:
2  T0 : PC(E) , MAR(E,L)
3  T1 : PC(I) , RAM(E) , MDR(L)
4
5  PC=01 IR=00 MDR=7E3 MAR=00 A=700 SP=FF B=781
6  (ION=0 C=0 Z=0 R=0) FLG=0
7  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
8  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
9
10 ~Decode:
11 This is non MRI Instruction
12 T2:  MDR(E) ,IR(L)          // IR <-MDR(4-7)
13
14 PC=01 IR=2E MDR=7E3 MAR=00 A=700 SP=FF B=781
15 (ION=0 C=0 Z=0 R=0) FLG=0
16 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
17 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
18
19 ~Execution:
20
21 * * * * * ASC A,3 * * * * *
22
23 PC=01 IR=2E MDR=7E3 MAR=00 A=0E0 SP=FF B=781
24 (ION=0 C=0 Z=0 R=0) FLG=0
25 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
26 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
27
28 T6:  A(E,L), BS(E), Reset

```

```

29 -- NEXT INSTRUCTION --
30
31 ~Fetch:
32 T0 : PC(E) , MAR(E,L)
33 T1 : PC(I) , RAM(E) , MDR(L)
34
35 PC=02 IR=2E MDR=7FF MAR=01 A=0E0 SP=FF B=781
36 (ION=0 C=0 Z=0 R=0) FLG=0
37 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
38 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
39
40 ~Decode:
41 This is non MRI Instruction
42 T2:  MDR(E) ,IR(L)          // IR <-MDR(4-7)
43
44 PC=02 IR=2F MDR=7FF MAR=01 A=0E0 SP=FF B=781
45 (ION=0 C=0 Z=0 R=0) FLG=0
46 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
47 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
48
49 * * * * * HALT has occured * * * * *
50
51 PC=02 IR=2F MDR=7FF MAR=01 A=0E0 SP=FF B=781
52 (ION=0 C=0 Z=0 R=0) FLG=0
53 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
54 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
55

```

3. SKI 2

Input File:-

Number of Instructions :- 4
SKI 2
ADD B
ASR 3
HLT

A screenshot of a text editor window titled 'input.txt'. The window has a dark background and a light-colored border. The content of the file is as follows:

Line	Instruction
1	4
2	F22
3	700
4	7E3
5	7FF

The line numbers 1 through 5 are listed on the left, and the corresponding instruction values are listed on the right. The line number 4 is highlighted with a light blue background.

Output File:-

◀ ▶ output.txt ✕

```
1 ~Fetch:
2 T0 : PC(E) , MAR(E,L)
3 T1 : PC(I) , RAM(E) , MDR(L)
4
5 PC=01 IR=00 MDR=F22 MAR=00 SP=FF A=E52 B=000
6 (ION=0 C=0 Z=0 R=0) FLG=0
7 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
8 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
9
10 ~Decode:
11 This is non MRI Instruction
12 T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
13
14 PC=01 IR=32 MDR=F22 MAR=00 SP=FF A=E52 B=000
15 (ION=0 C=0 Z=0 R=0) FLG=0
16 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
17 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
18
19 ~Execution:
20
21 * * * * * SKI 2 * * * * *
22
23 T6: PC(I), Reset
24
25 PC=02 IR=32 MDR=F22 MAR=00 SP=FF A=E52 B=000
26 (ION=0 C=0 Z=0 R=0) FLG=0
27 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
28 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
29
```

```
30
31 --- NEXT INSTRUCTION ---
32
33 ~Fetch:
34 T0 : PC(E) , MAR(E,L)
35 T1 : PC(I) , RAM(E) , MDR(L)
36
37 PC=03 IR=32 MDR=7E3 MAR=02 SP=FF A=E52 B=000
38 (ION=0 C=0 Z=0 R=0) FLG=0
39 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
40 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
41
42 ~Decode:
43 This is non MRI Instruction
44 T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
45
46 PC=03 IR=2E MDR=7E3 MAR=02 SP=FF A=E52 B=000
47 (ION=0 C=0 Z=0 R=0) FLG=0
48 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
49 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
50
51 ~Execution:
52
53 * * * * * ASR A,3 * * * * *
54
55 PC=03 IR=2E MDR=7E3 MAR=02 SP=FF A=FCA B=000
56 (ION=0 C=0 Z=0 R=0) FLG=0
57 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
58 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
59
60 T6: A(E,L), BS(E), Reset
61
```

```

62  --- NEXT INSTRUCTION ---
63
64  ~Fetch:
65  T0 : PC(E) , MAR(E,L)
66  T1 : PC(I) , RAM(E) , MDR(L)
67
68  PC=04 IR=2E MDR=7FF MAR=03 SP=FF A=FCA B=000
69  (ION=0 C=0 Z=0 R=0) FLG=0
70  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
71  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
72
73  ~Decode:
74  This is non MRI Instruction
75  T2: MDR(E) ,IR(L)          // IR <-MDR(4-7)
76
77  PC=04 IR=2F MDR=7FF MAR=03 SP=FF A=FCA B=000
78  (ION=0 C=0 Z=0 R=0) FLG=0
79  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
80  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
81
82  * * * * * HALT has occured * * * * *
83
84  PC=04 IR=2F MDR=7FF MAR=03 SP=FF A=FCA B=000
85  (ION=0 C=0 Z=0 R=0) FLG=0
86  IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
87  OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
88
89

```


Interrupt Handling

It is implemented by kbhit() whenever any key is pressed it checks whether the key is "i". If the key pressed is "i" then an interrupt occurs and sets the R flag. It will jump to Interrupt handling Subroutine and the return address is saved in the stack.

Interrupt handling Subroutine is present from 0xF0 location in RAM and at last location of subroutine has a JMP instruction which takes the PC to the main program.

After interrupt handling, the PC takes the return address from the stack and resets the R flag.

There is an ION flag which signifies whether a processor can take an interrupt or not.

ION flag = 0 indicates that the processor can take and handle Interrupt.

ION flag = 1 indicates that the processor won't take and handle any Interrupt.

Example:

Input File:-

Number of Instructions :- 3

ADD B

SUB B

HLT

```
no of instructions:
3
700
710
7FF
```

Output File:-

```

6 ~Fetch:
7 T0 : PC(E) , MAR(E,L)
8 T1 : PC(I) , RAM(E) , MDR(L)
9
10 PC=01 IR=00 MDR=700 MAR=00 SP=FF A=700 B=000
11 (ION=0 C=0 Z=0 R=0) FLG=0
12 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
13 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
14
15 ~Decode:
16 This is non MRI Instruction
17 T2: MDR(E) ,IR(L) // IR <-MDR(4-7)
18
19 PC=01 IR=20 MDR=700 MAR=00 SP=FF A=700 B=000
20 (ION=0 C=0 Z=0 R=0) FLG=0
21 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
22 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
23
24 ~Execution:
25
26 * * * * * ADD B * * * * *
27
28 T3: Idle
29 T4: Idle
30 T5: A(E,L) , B(E) , ALU(000) , Reset
31
32 PC=01 IR=20 MDR=700 MAR=00 SP=FF A=700 B=000
33 (ION=0 C=0 Z=0 R=1) FLG=1
34 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
35 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
36
37 * * * * * Interrupt Has Occured* * * * *
38 PC jump to location 0xF0 where Interrupt handling subroutine is written and Stack has the return address and R flag is set.
39
40 PC=F0 IR=20 MDR=700 MAR=FE SP=FE A=700 B=000
41 (ION=0 C=0 Z=0 R=1) FLG=1
42 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
43 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000

```

```

44
45 .....Interrupt Handling.....
46 .
47 .
48 .
49 .
50 .
51 .
52 .
53 .
54 * * * * * Interrupt has been serviced* * * * *
55
56 PC=F1 IR=20 MDR=700 MAR=FE SP=FE A=700 B=000
57 (ION=0 C=0 Z=0 R=0) FLG=0
58 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
59 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
60
61 * * * * * PC will jump to main program where the interrupt has occurred* * * * *
62
63 PC=01 IR=20 MDR=700 MAR=FE SP=FF A=700 B=000
64 (ION=0 C=0 Z=0 R=0) FLG=0
65 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
66 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
67
68
69 --- NEXT INSTRUCTION ---
70
71 ~Fetch:
72 T0 : PC(E) , MAR(E,L)
73 T1 : PC(I) , RAM(E) , MDR(L)
74
75 PC=02 IR=20 MDR=710 MAR=01 SP=FF A=700 B=000
76 (ION=0 C=0 Z=0 R=0) FLG=0
77 IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
78 OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000
79

```

```

~Decode:
This is non MRI Instruction
T2:  MDR(E) ,IR(L)          // IR <-MDR(4-7)

PC=02 IR=21 MDR=710 MAR=01 SP=FF A=700 B=000
(ION=0 C=0 Z=0 R=0) FLG=0
IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000

~Execution:

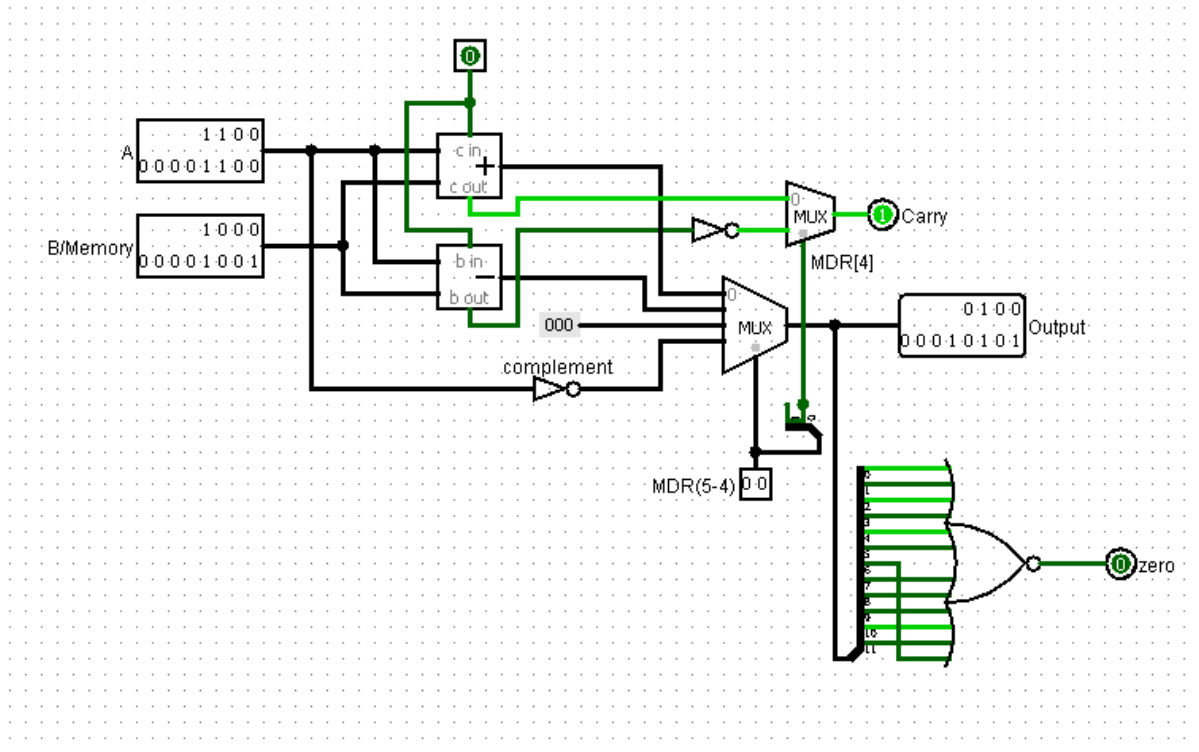
PC=03 IR=2F MDR=7FF MAR=02 SP=FF A=0 B=000
(ION=0 C=0 Z=0 R=0) FLG=0
IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000

* * * * * * * HALT has occurred * * * * * * *

PC=03 IR=2F MDR=7FF MAR=02 SP=FF A=0 B=000
(ION=0 C=0 Z=0 R=0) FLG=0
IN1=005 IN2=000 IN3=000 IN4=000 -- FGI=0110
OUT1=000 OUT2=000 OUT3=000 OUT4=000 -- FGO=0000

```

Hardware implementation of ALU



Based on values of MDR[5] and MDR[4], result is selected among 4 different operations.

00 - ADD

01 - SUB

10 - CLA

11 - CMA

And correspondingly zero and carry flags will be updated.

Updating Carry Flag :

In addition, based on the values of A and (B or from memory) if calculation happens to generate a carry then carry flag will be set i.e. 1 otherwise 0.

In case of Subtraction, we are using a hardware which computes subtraction as:

$A \text{ SUB } B = A - B + C - 1$, where C is carry flag

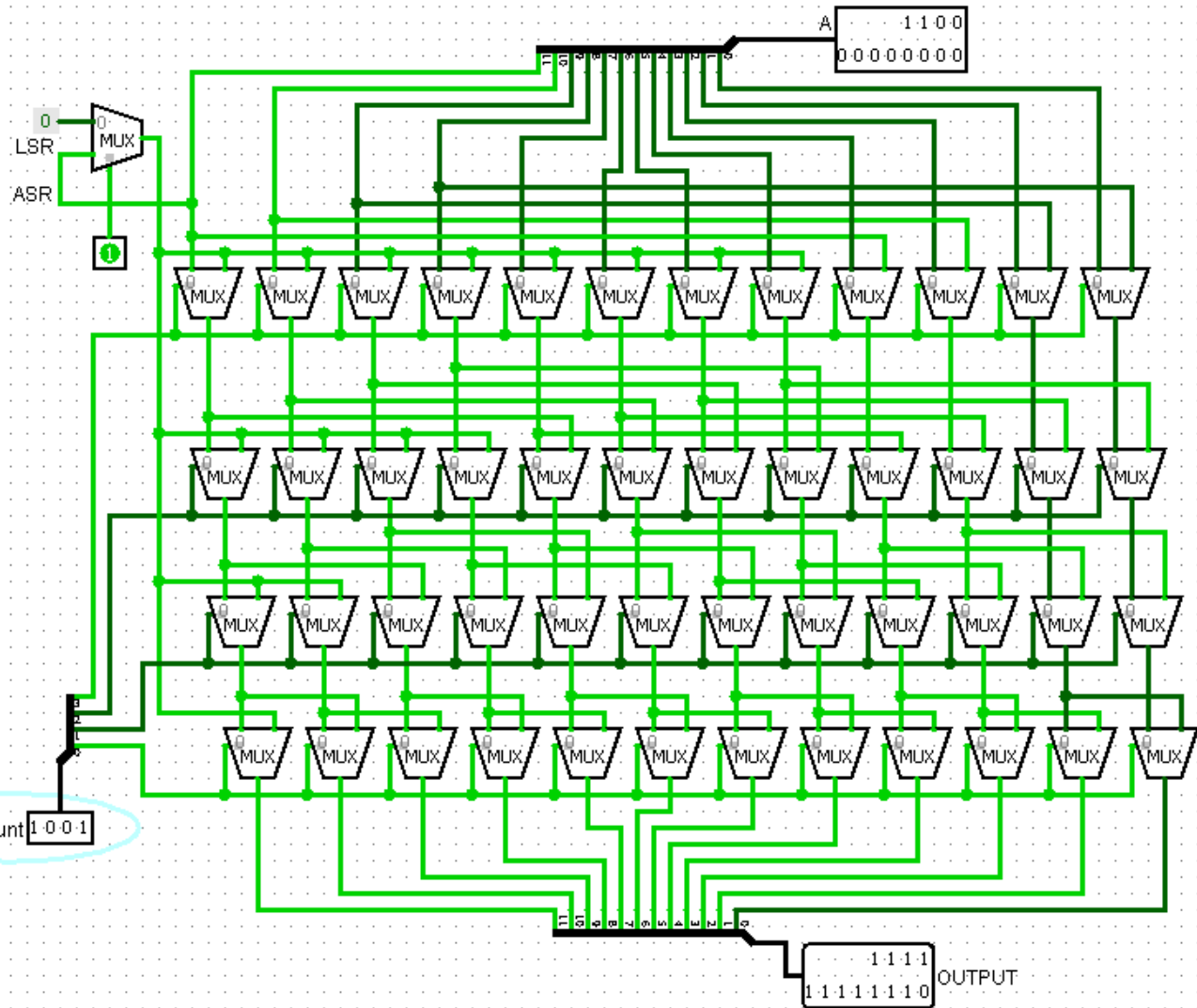
If $C == 1$ (set), simply $A - B$

else if $C == 0$, then $A - B - 1$ and hence use of NOT gate in b/w is justified.

Updating Zero Flag :

The NOR gate outputs a 1 only if all input bits are 0. If any input bit is a 1, the NOR's output is a 0.

Hardware implementation of ASR in Barrel Shifter



If $A[11] = 1$ i.e. value in accumulator is negative and hence shifted bits are '1'.
 If $A[11] = 0$ i.e. value in accumulator is positive and hence shifted bits will be '0'.

Barrel shifter is able to complete the shift in a single clock cycle, giving it a great advantage over a simple shifter which can shift n bits in n clock cycles.

Conclusion

EDITH deals with instruction set of size 12-bit using RAM 256X12 bits. It decodes a macro-instruction as I bit, Opcode(3-bits) and 8-bits for address. MSB is considered as I bit, which signifies the type of addressing whether direct or indirect while dealing with MRI instructions or whether register type or I/O type instructions in case of NON-MRI instructions.

MRI : Next to I bit, 3 bits are used for opcode which is used to identify the type of instruction we want to execute in a program, followed by last 8-bits that are used for address handling.

NON-MRI : Next to I bit, 3 bits are always '111', further 4 bits are identifying as opcode for non-MRI instructions, it signifies the type of instruction to be executed, and last 4 bits represents data in case of MVI instruction, shift count in case of shift operations, port selector in case of I/O operation, and can be any value otherwise.

Flags dealt in the Processor are ION, C , Z , R

ION: To signify whether a processor can take an interrupt or not.

C: To signify the carry bit while dealing the arithmetic instructions.

Z: To signify the zero value of the accumulator

R: To signify an interrupt.

EDITH also takes care of an interrupt (**CONDITION : when ION = 0**) that might be generated while executing a program. It first completes the execution of the particular macro-operation it is performing and then handles interrupt and further it continues to execute the program where it has left off.