Military Institute of Science and Technology

4-BIT MICROPROCESSOR

Course Name: Digital System Design Sessional

Course Code: CSE-316

Program: CSE-20

Session: 2019-2020

Group No: B3

Group members:

Jamal Uddin Tanvin (202014016)

MD Rifat Islam (202014034)

Md Tausiful Haque (202014036)

Nurshat Fateh Ali (202014040)

TABLE OF CONTENTS

4-Bit Microprocessor

1. Introduction

1.1. Simple 4 Bit Micro Processor (SAP-1):

A 4 Bit Microprocessor is a processor that can processes 4-bit number simultaneously. The size of number that the processor can work with depends on the size of the ALU (Arithmetic Logic Unit). An n-bit microprocessor has an ALU of n bit. We can say that the size of ALU of a 4-bit microprocessor is 4 bits. SAP is simple microprocessor with ALU of 8 bits in length. In this project we will design a simple 4-bit microprocessor.

The Simple-As-Possible (SAP) computer is a simplified computer architecture designed for educational purposes. The SAP architecture serves as an example in Digital Computer Electronics for building and analyzing complex logical systems with digital electronics. Digital Computer Electronics successively develops three versions of this computer, designated as SAP-1, SAP-2, and SAP-3. Each of the last two build upon the immediate previous version by adding additional computational, flow of control, and input/output capabilities. SAP-2 and SAP-3 are fully Turing-complete. [1]

The SAP-1 computer is the first stage in this evolution and contains the necessities for a functional computer. Its primary purpose is to develop a basic understanding of how a computer works, interacts with memory and other parts of the system like input and output. The instruction set is very limited and is simple.

1.2. Macro instruction set:

A macro instruction is a request to the assembler program to process a predefined sequence of instructions called a macro definition. From this definition, the assembler generates machine and assembler instructions, which it then processes as if they were part of the original input in the source module. [2] It is a group of programming instructions that have been compressed into a simpler form and appear as a single instruction.

In SAP the control sequencer sends out control words, one during each T state or clock cycle. These words are like directions telling the rest of the computer what to do. Because they produce small steps in the data processing where each control word is called a **microinstruction**. The instructions we have been programming with (LDA, SUB, MOV…) are called **macroinstructions** to distinguish them from microinstructions. In SAP-1 each macroinstruction is consist of three microinstructions. In this project, we have designed the macroinstructions in such a way that they consist of 2 microinstructions each and, we have few macroinstructions which are not in SAP-1. The following macroinstructions are implemented in our project.

LDA, MOV, INR, AND, JM, OUT and HALT

1.3. Programming SAP-1:

In SAP-1 we have LDA, ADD, SUB, OUT, HLT these macroinstructions. With these instructions we can easily make a program in Assembly language and then convert them in Machine code to write in the RAM. In SAP-1 we manually write the program in the RAM and execute it by giving clock pulse manually (Single Step) or automatically. In our project we also program our microprocessor same as SAP-1. In our case we have different instructions than SAP-1 but the process of programming is same. A simple program in SAP-1 is like below

10H – 22H + 28H

Memory Addresses (RAM)	Assembly code	Machine code
00H	LDA 8H	0000 1000
01H	SUB 9H	0010 1001
02H	ADD AH	0001 1011
03H	OUT	1110 xxxx
04H	HLT	1111 xxxx
05H		
06H		
07H		
08H	10H	0001 0000
09H	22H	0010 0010
0AH	28H	0010 1000
0BH		
0CH		
0DH		
0EH		
0FH		

Figure 1: Simple program in SAP-1

2. Architecture

The architecture of SAP-1 is a bus organized computer. The architecture of our microprocessor project is same as SAP-1.

2.1. Block diagram:

The block diagram of the architecture our microprocessor project is given further for better understanding.

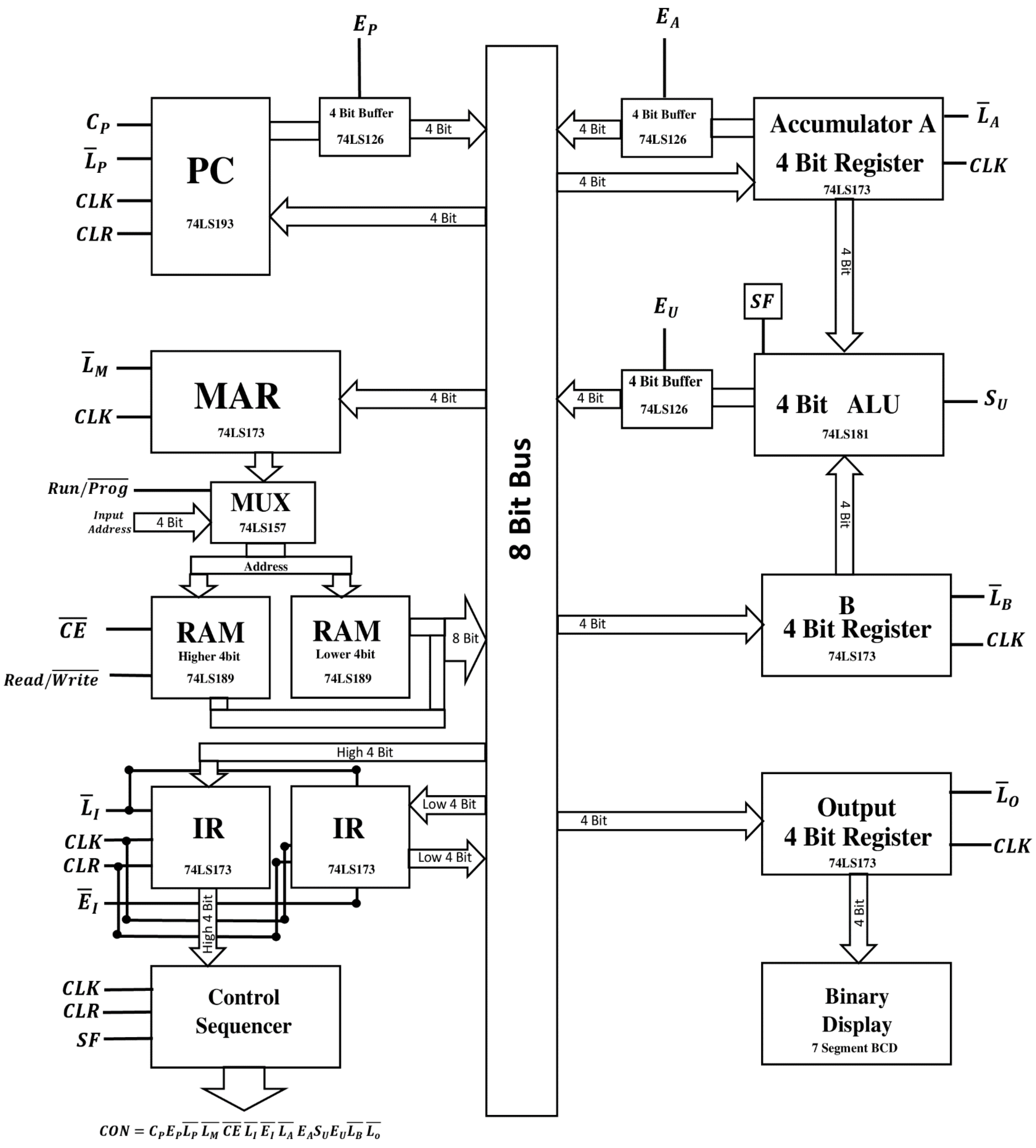

Figure 2: Block diagram of the Architecture

3. Fetch cycle and active control signals

In our project we have total four T states. The fetch cycle has two T state, T_1 and T_2 as we have merged the increment state with memory state. So T_1 does the job address state and T_2 state does the job of both increment and memory state. The control signal state during fetch cycle is given below

Control word format:

$$\boldsymbol{CON = C_P E_P \overline{L_P}\,\overline{L_M}\,\overline{CE}\,\overline{L_I}\,\overline{E_I}\,\overline{L_A}\,E_A S_U E_U \overline{L_B}\,\overline{L_o}}$$

$\boldsymbol{C_P}$ = Increase Program counter

$\boldsymbol{E_P}$ = Enable Program counter

$\boldsymbol{\overline{L_P}}$ = Load program counter

$\boldsymbol{\overline{L_M}}$ = Load MAR

$\boldsymbol{\overline{CE}}$ = Enable RAM

$\boldsymbol{\overline{L_I}}$ = Load IR

$\boldsymbol{\overline{E_I}}$ = Enable IR

$\boldsymbol{\overline{L_A}}$ = Load Accumulator

$\boldsymbol{E_A}$ = Enable Accumulator

$\boldsymbol{S_U}$ = Increment (0) / AND (1)

$\boldsymbol{E_U}$ = Enable ALU

$\boldsymbol{\overline{L_B}}$ = Load B register

$\boldsymbol{\overline{L_o}}$ = Load Output register

Macro	**State**	**CON(HEX)**	**Active**
Fetch	T_1	DE3	E_P , $\overline{L_M}$
	T_2	1663	$C_P, \overline{CE}, \overline{L_I}$

4. Execution cycle and active control signals

Among four T states in our project T_3 and T_4 states are for the execution cycle. The register transfer during the execution cycle depend on the particuler instruction being executed. In our project we have total 7 instructions. They are given below

Serial	OP code	Instruction	Function
1	0000	LDA M	Load RAM data from memory address M to Accumulator
2	0001	MOV B, A	Move the 4-bit value to B register from A register
3	0010	INR A	Increment the value of Accumulator by 1
4	0011	AND B	Perform AND operation on data of B register with accumulator and store the result in the A register
5	0100	JM M	Jump to designated memory address M if SF=1
6	0101	OUT	Load data of Accumulator to output register
7	0110	HLT	Stop the program (No operation)

The control signal state during execution cycle for each instruction is given below

Macro	State	CON(HEX)	Active
LDA	T_3	05A3	$\overline{E_I}, \overline{L_M}$
	T_4	06C3	$\overline{CE}, \overline{L_A}$
MOV	T_3	07F1	$E_A, \overline{L_B}$
	T_4	07E3	NOP
INR	T_3	07C7	$E_U, \overline{L_A}$
	T_4	07E3	NOP
AND	T_3	07CF	$S_U, E_U, \overline{L_A}$
	T_4	07E3	NOP
JM When SF=1	T_3	03A3	$\overline{E_I}, \overline{L_P}$
	T_4	07E3	NOP
OUT	T_3	07F2	$E_A, \overline{L_o}$
	T_4	07E3	NOP
HALT	T_3	07E3	NOP
	T_4	07E3	NOP

5. Design of execution Unit

5.1. Program counter, RAM, ALU:

In our project we have used IC 74LS193 as our program counter. IC 74LS193 is a synchronous 4-Bit Up Down Binary Counter with Dual clock.

For RAM (Random Access Memory) we have used IC 74LS189. As 74LS189 is a 16×4 RAM but we needed store 8-bit data, so we took two 16×4 RAM and shorted their address bit and made 16×8 RAM with them.

For ALU (Arithmetic Logic Unit) we used IC 74LS181. As it offers lots of operations, we just used two operations (Increment, AND) toggling the two combinations of the selection bits with $\boldsymbol{S_U}$ **(**0 and 1).

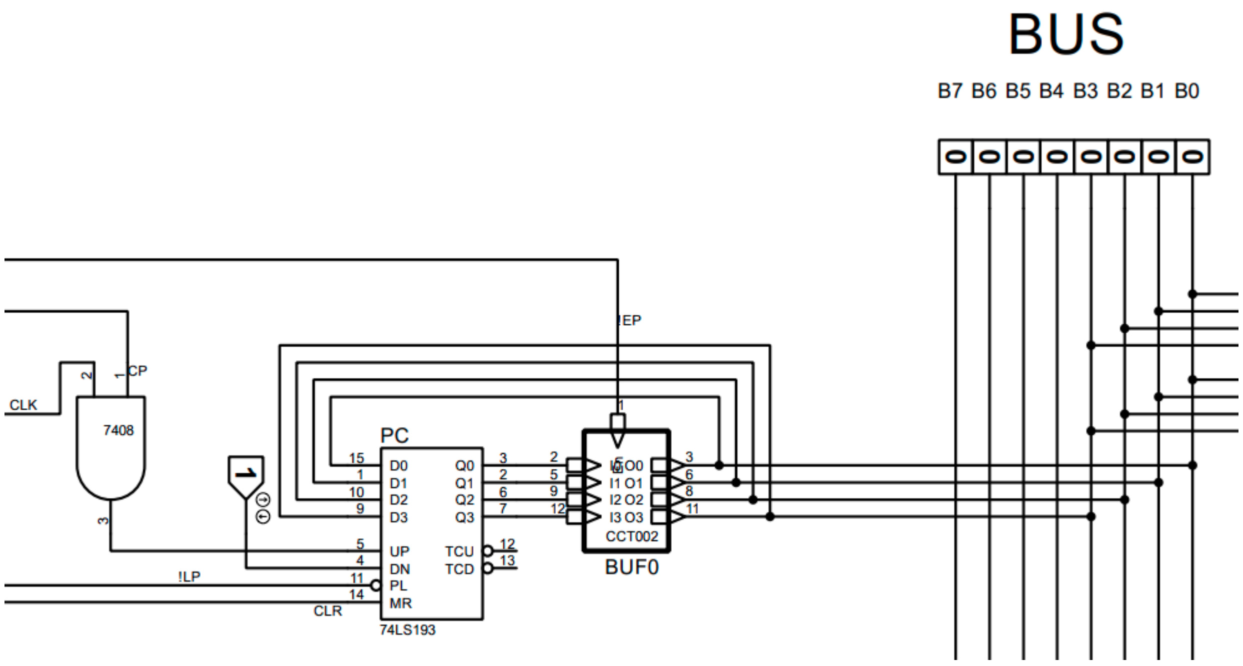

Figure 3: Design of Program Counter

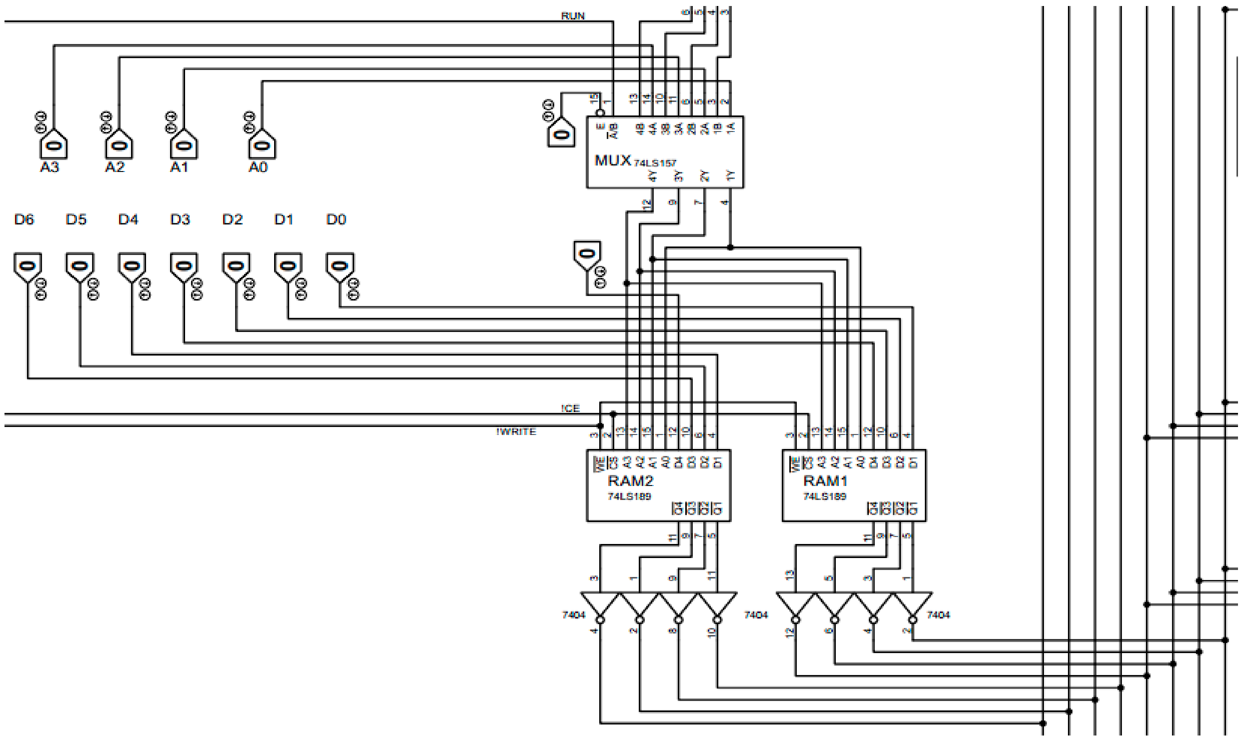

Figure 4: Design of RAM

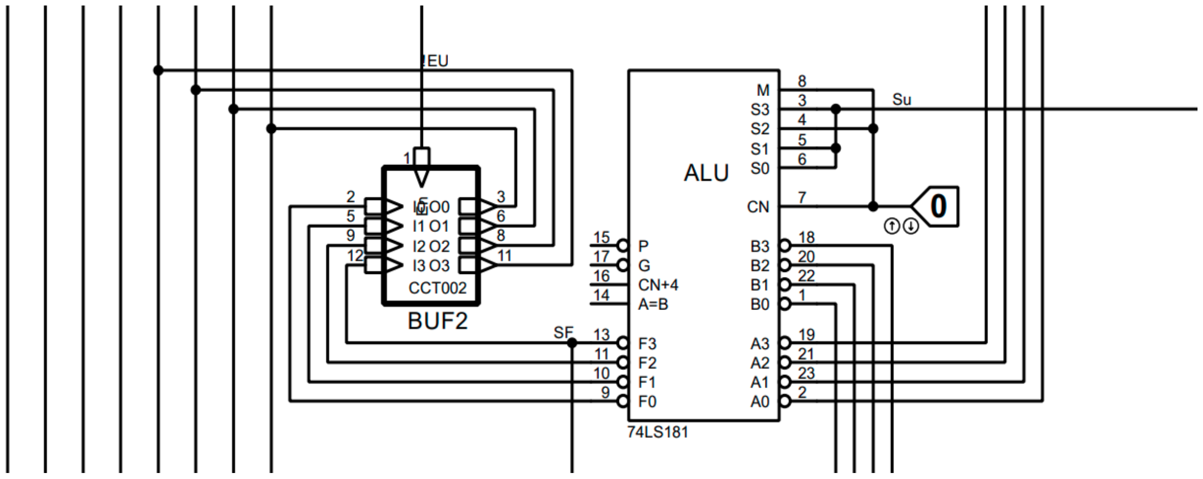

Figure 5: Design of ALU

5.2. Registers (A, B, OUT, MAR, IR):

For register we have used IC 74LS173 which is a 4 bit register of Quad D-Type Flip-Flops with Tristate Outputs for Accumulator, B register, Output register, MAR, and IR. We used two registers for IR as it is of 8 bits in size.

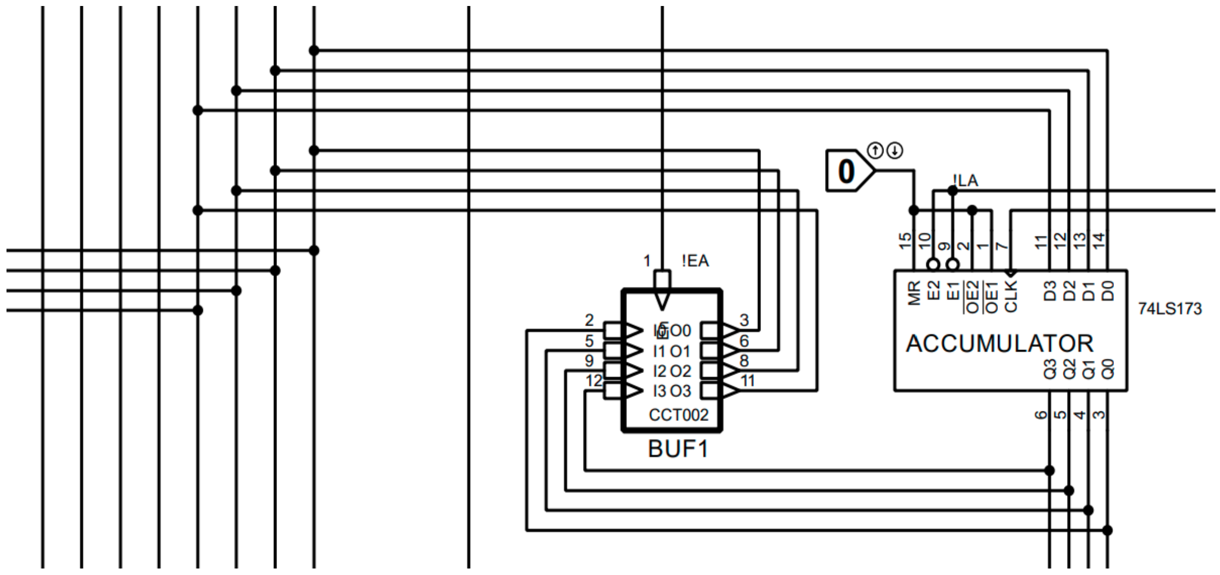

Figure 6: Design of Accumulator

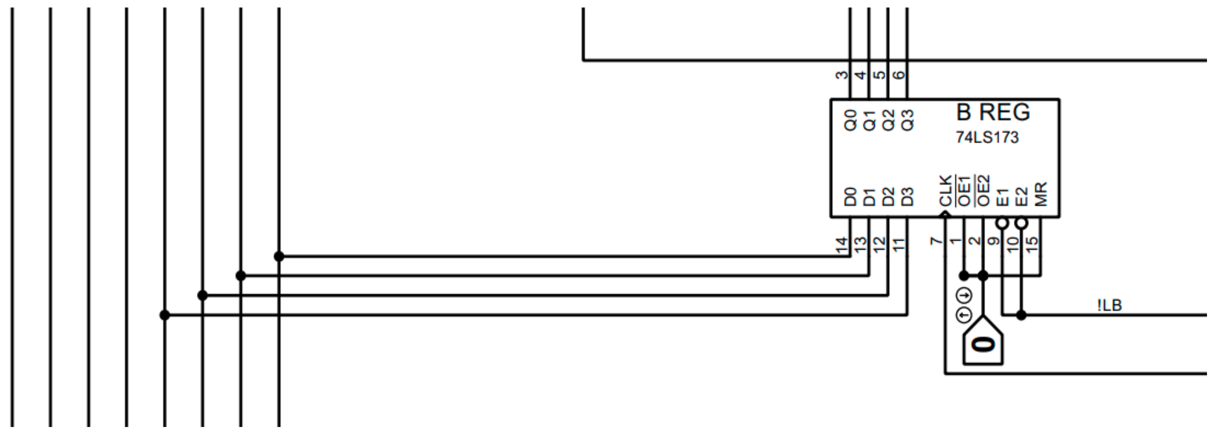

Figure 7: Design of B register

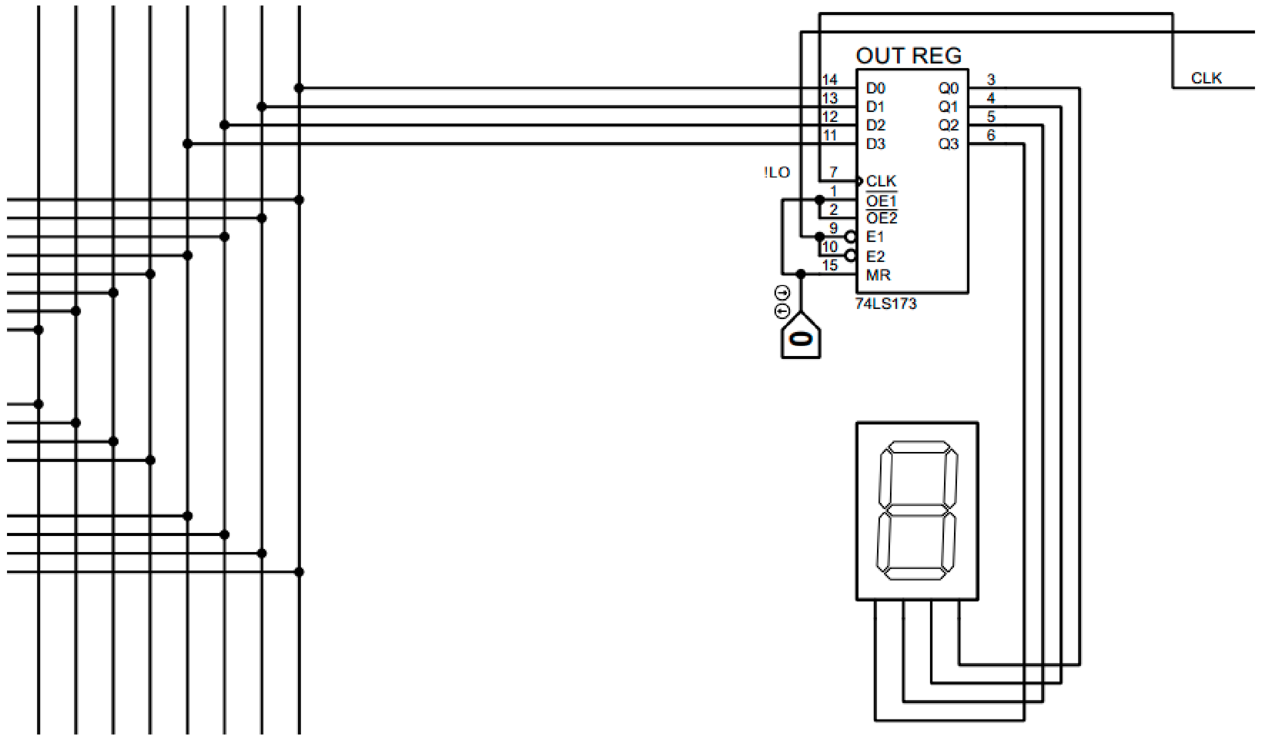

Figure 8: Design of Output register

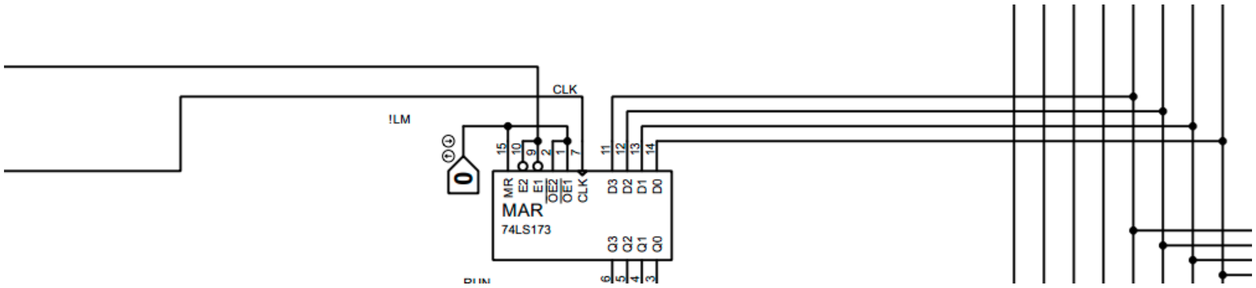

Figure 9: Design of MAR

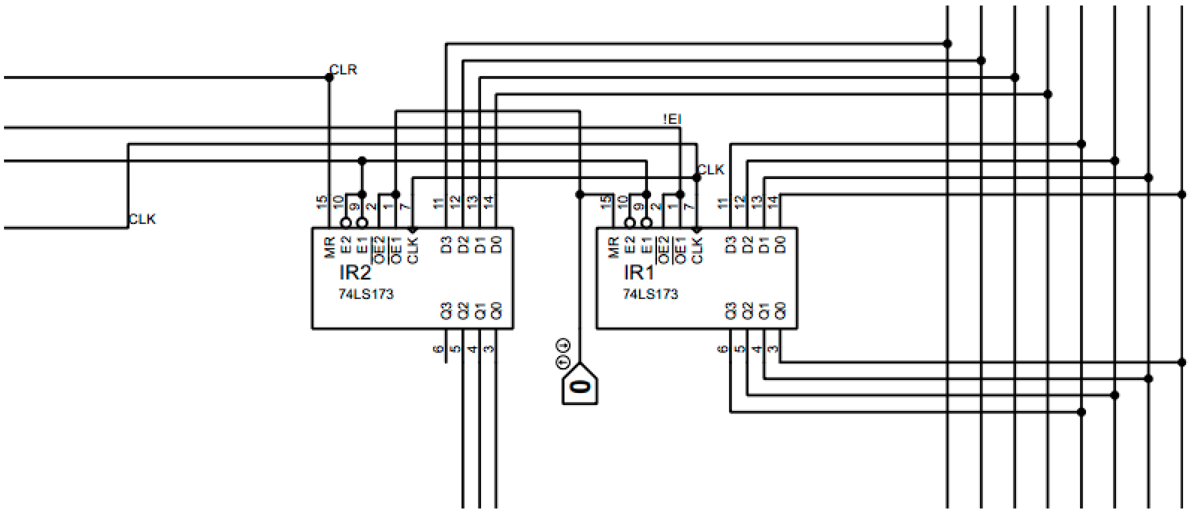

Figure 10: Design of IR

5.3. MUX and Buffers:

We used IC 74LS157 which is Quadruple 1-of-2 Data Selectors/Multiplexers as our MUX. MUX is required for giving Address in the RAM from outside od the processor while programing the microprocessor. We used IC 74LS125 which is a Quadruple Bus Buffer Gate with Tristate Outputs for holding data. Buffer is used to convert an IC into 3 state from 2 state.

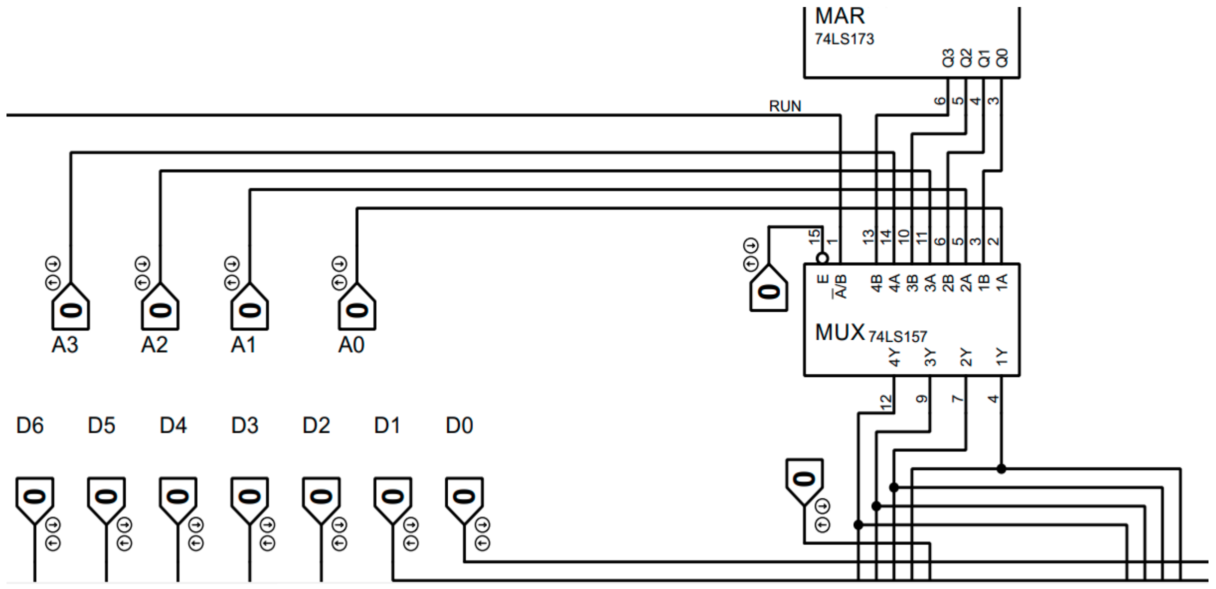

Figure 11: Design of MUX

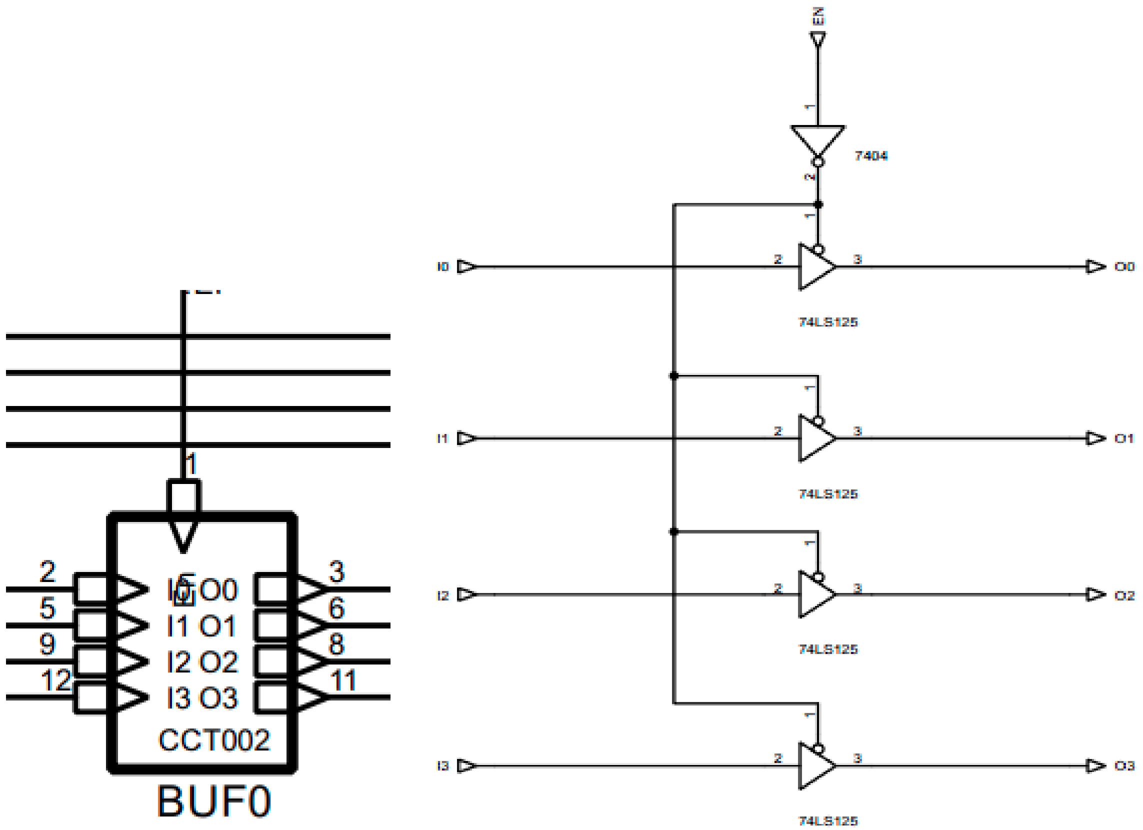

Figure 12: Design of BUFFER

5.4. Required ICs for Execution Unit:

IC	IC type	Function	Quantity
74LS193	Synchronous 4-Bit Up Down Binary Counter with Dual clock	Program Counter	1
74LS189	64-Bit Random Access Read/Write Memory	RAM	2
74LS181	Arithmetic Logic Unit	ALU	1
74LS173	Quad D-Type Flip-Flops with Tristate Outputs	4 Bit Register	6
74LS157	Quadruple 1-of-2 Multiplexers	MUX	1
74LS125	Quadruple Bus Buffer Gate	BUFFER	3
74LS04	NOT Gate	Inverter	2
74LS08	AND Gate	AND	1
Total			17

6. Design of Control Unit

6.1. Ring counter:

By the help of ring counter, we change our T states during fetch and execution cycle. We need a ring counter which can enable the states sequentially in a circular way. So, we made a Ring counter with JK flipflops (IC 74LS76) to do that job.

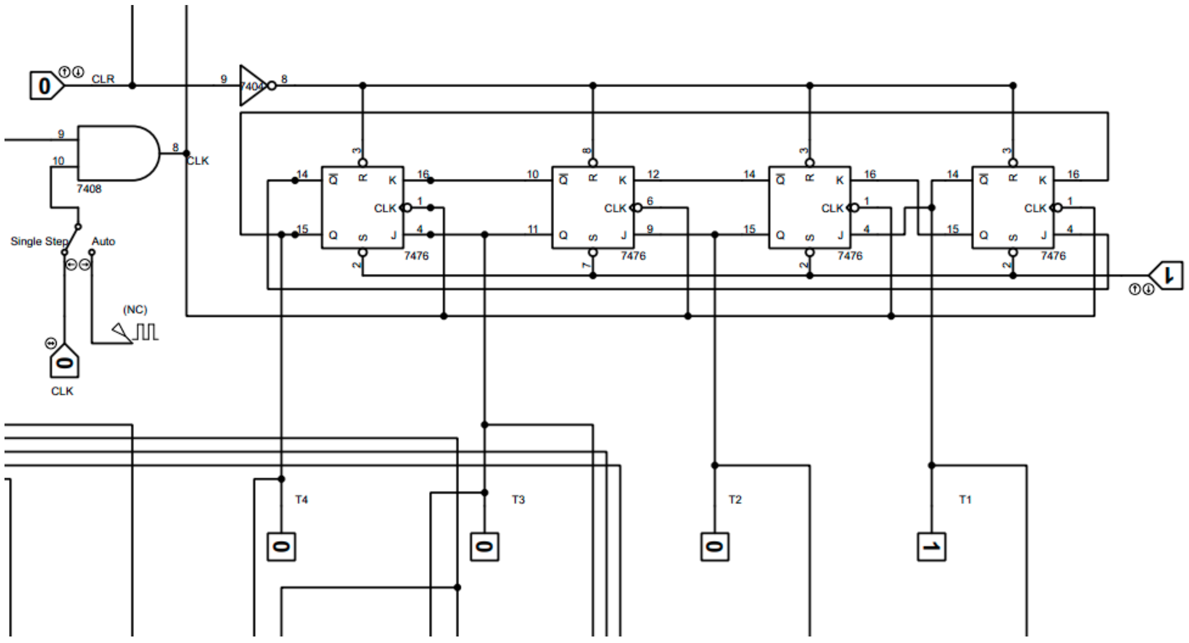

Figure 13: Design of Ring counter

6.2. Instruction decoder:

We must decode the instruction written in the RAM after fetching them after the fetch cycle to continue the process of our execution cycle. To do this we used IC 74LS138 which is a 3-Line to 8-Line Decoders/Demultiplexers. It decodes the opcodes and enables the corresponding line for that opcode.

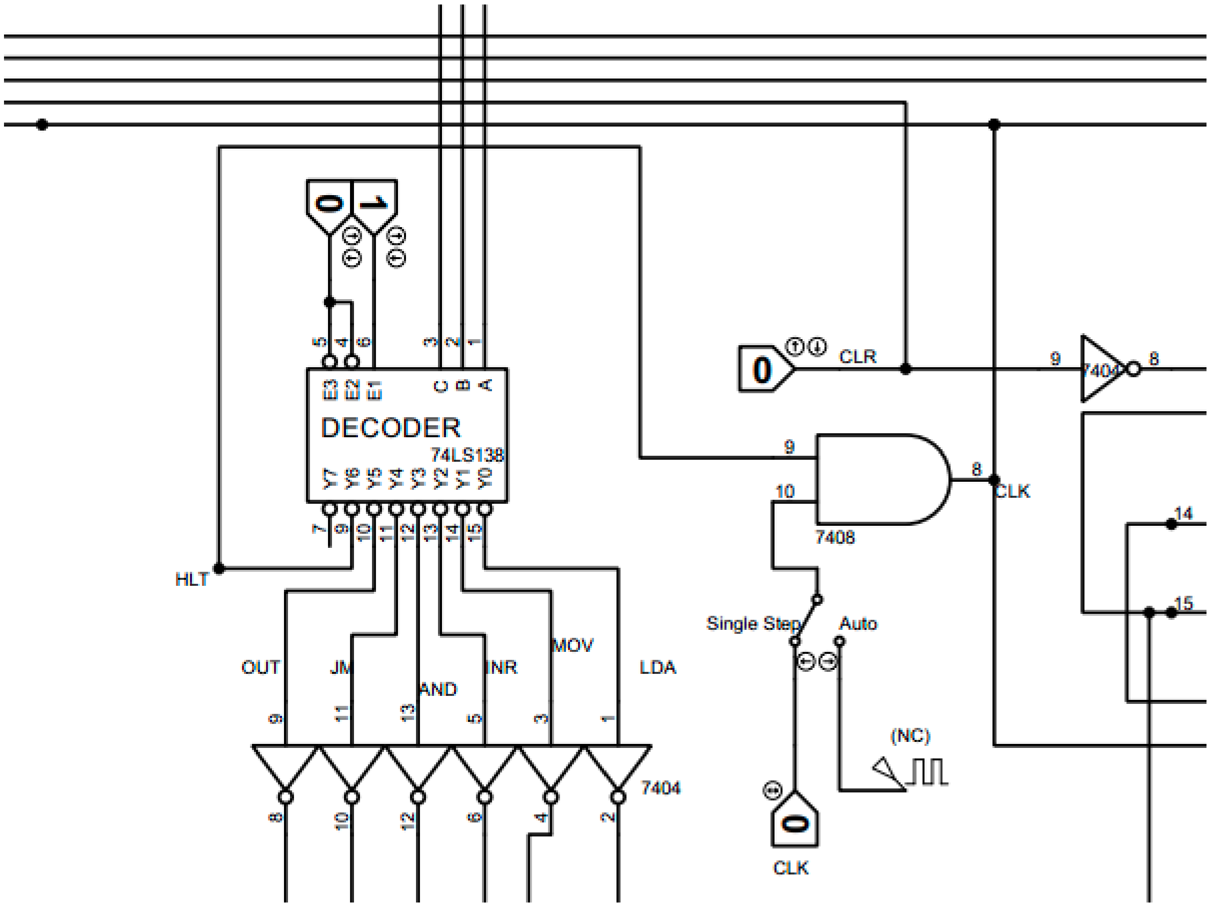

Figure 14: Design of Decoder

6.3. Control Matrix:

Control matrix does the job of enabling or disabling the control signals of the execution unit. We have designed the control matrix with combinational circuit consist of AND, OR, NOR, and NOT gates. The functions for the signals are given below

$C_P = T_2$

$E_P = T_1$

$\overline{L_P} = \overline{T_3 . JM . SF}$

$\overline{L_M} = \overline{T_1 + T_3.LDA}$

$\overline{CE} = \overline{T_2 + T_4.LDA}$

$\overline{L_I} = \overline{T_2}$

$\overline{E_I} = \overline{T_3.LDA + T_3.JM.SF}$

$\overline{L_A} = \overline{T_4.LDA + T_3.INR + T_3.AND}$

$E_A = T_3.MOV + T_3.OUT$

$S_U = T_3.AND$

$E_U = T_3.INR + T_3.AND$

$\overline{L_B} = \overline{T_3.MOV}$

$\overline{L_o} = \overline{T_3.OUT}$

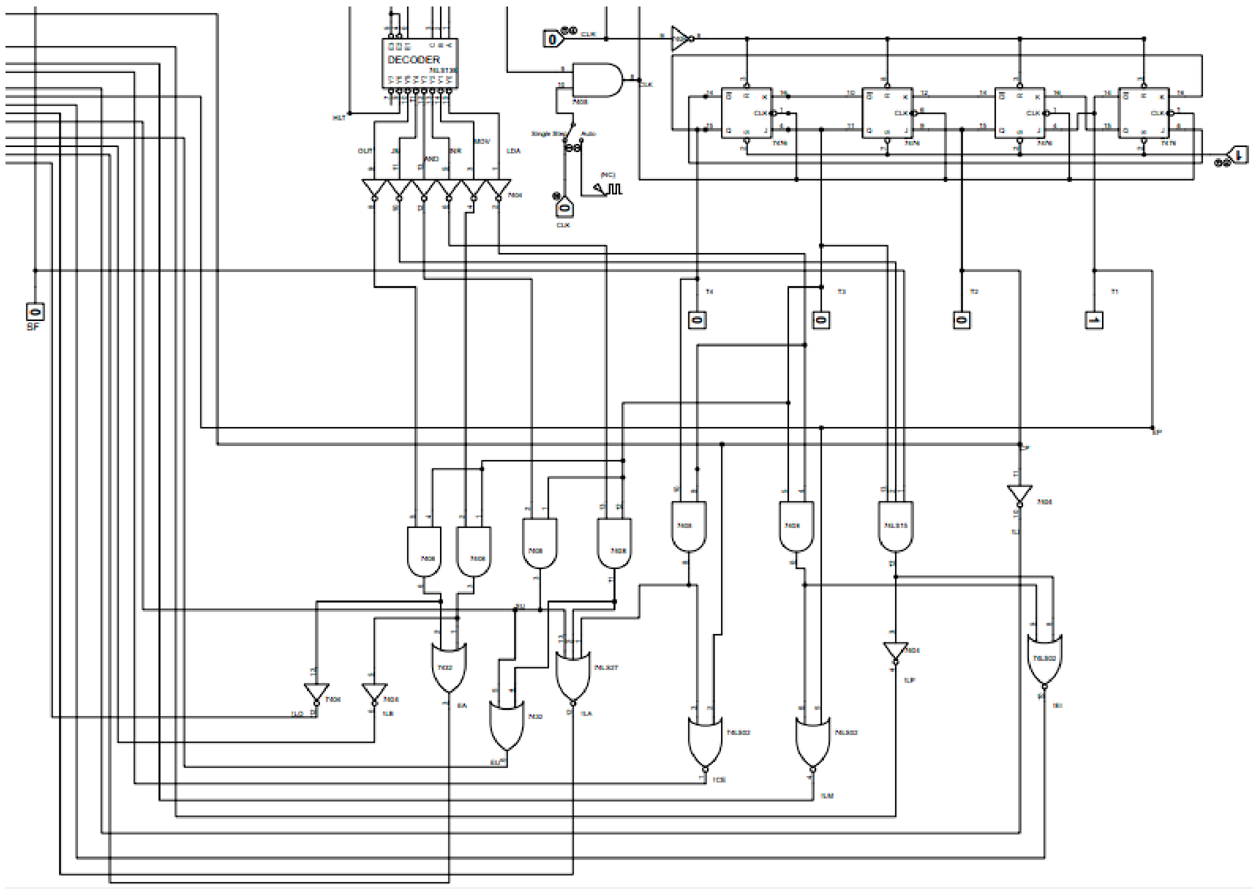

Figure 15: Design of Control Matrix

6.4. Required ICs for Control Unit:

IC	IC type	Function	Quantity
74LS138	3-Line To 8-Line Decoders	Decoding	1
74LS76	JK Flipflop	Ring Counter	2
74LS04	NOT Gate	Inverter	2
74LS08	AND Gate	AND operation	2
74LS32	OR Gate	OR operation	1
74LS32	NOR Gate	NOR operation	1
Total			9

7. Problem faced

In this project of 4 Bit Microprocessor, we faced some significant problems. Although all the problem had been overcome by us. Some are given below

- The trainer boards didn't have enough switch for all the signals and data inputs. So, we had to give those inputs by connecting wires manually to the ground or to the Vcc.
- We didn't have 16×8 RAM, so we had to use two 16×4 RAM. Again, the RAM gives inverted outputs. So, we had to reinvert the outputs of the RAMs.
- The decoder that we use also gives inverted output. So, we had to reinvert those also.
- To do only two operations (Increment & AND) we had to manipulate the selection pin of the ALU so that we can do these two operations with only one control signal.
- There were lots of wires to be put on in the ICs, so that was a hard job.
- In this project we needed total 26 ICs. As a result, we needed two trainer boards. One for the execution unit and another for the control unit.
- As we used Ram so every time the power went of all the data written in it was destroyed. So, we had to write instructions repeatedly and that was quite time consuming.

8. Future expansion

The microprocessor that we designed can easily be expended in future by adding new control signals and with some minor modification. But there is a problem. The control unit that we designed is Hardwired. So, it is quite too tough to change the design when a new instruction is implanted for the processor. We may have to change the whole control sequencer for that job. If we had a chance to make the control unit with microprogram method than it could easily expanded.

The elaborated design of the project

We designed our project using Proteus 8 professional version 8.13 SP0. The pdsprj and pdf file of the design can be found in this link:

https://github.com/Tanvin420/4-Bit_Microprocessor/

REFERENCES

[1] "Wikipedia,"[Online].Available:https://en.wikipedia.org/wiki/Simple-AsPossible_computer.

[2] "IBM,"[Online].Available:https://www.ibm.com/docs/en/zos/2.2.0?topic=SSLTBW_2.2.0/com.ibm.zos.v2r2.asma400/asmr102115.html.