ISLAMIC UNIVERSITY OF TECHNOLOGY

(IUT)

COURSE NO: EEE 4308 (DIGITAL ELECTRONICS LAB)

NAME OF THE PROJECT:

SAP-1 ARCHITECTURE-BASED 8-BIT COMPUTER DESIGN AND IMPLEMENTATION.

NAME OF THE TEAM:

THE "A" TEAM

TEAM MEMBERS:

I) **NAME:** REDWAN-UL-BARI

ID: 190021119

II) **NAME:** SHIHAB SHAHRIAR

ID: 190021123

III) **NAME:** SAFWAN AHMED

ID: 190021114

OBJECTIVE OF THE PROJECT

The primary objective of this project was 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.

Another objective of the project was to design the SAP-1 in such a way so that through designing we could implement the combinational and sequential logic circuits in our practical life. Besides, we also came to learn more about Proteus, how to use the components accordingly and observed the simulation results according to given instruction sets.

SAP-1 DESIGN

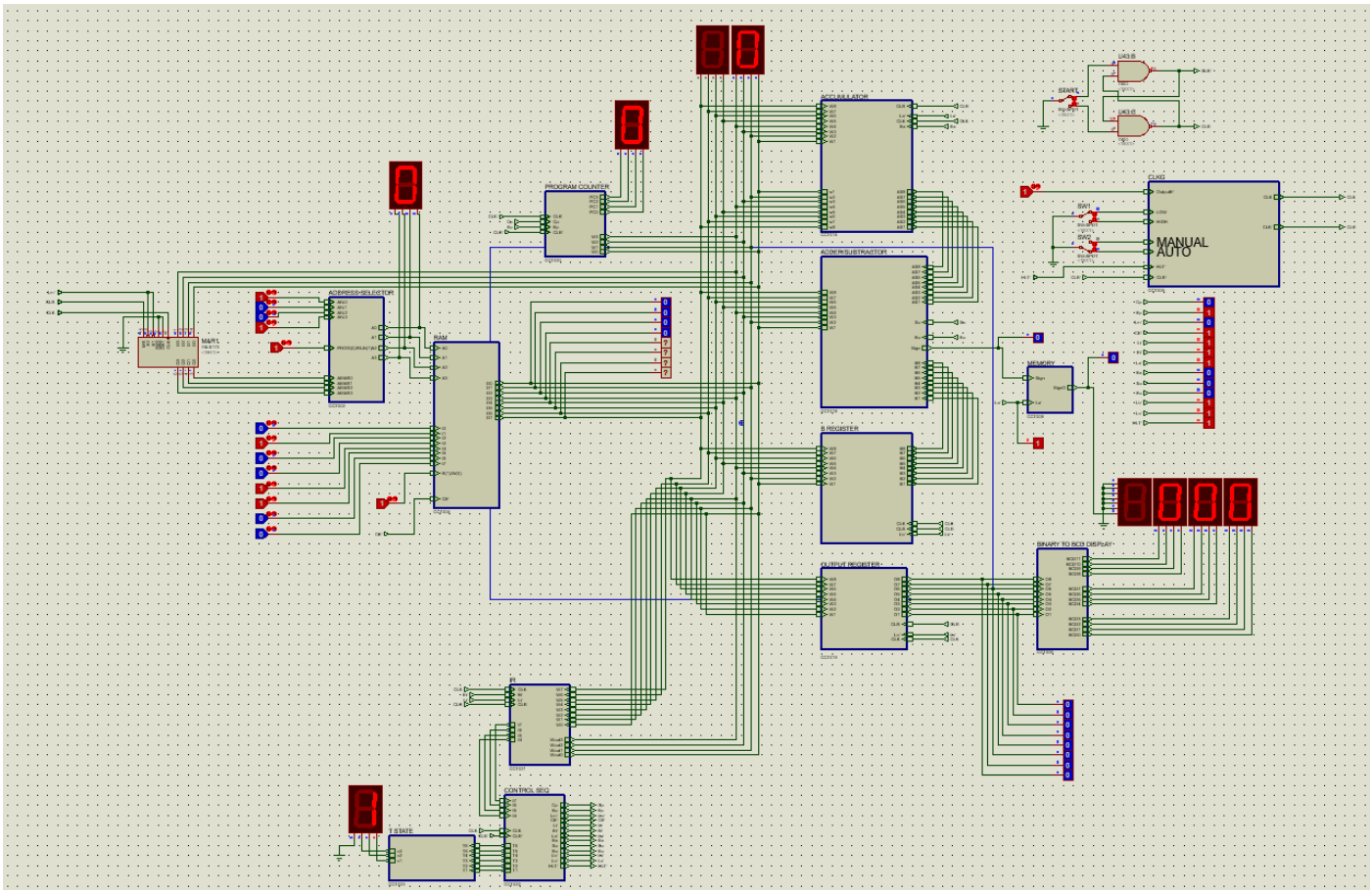


FIG: Complete block diagram of SAP-1

MAIN COMPONENTS OF SAP-1

1. MEMORY ADDRESS REGISTER (MAR)
2. ADDRESS SELECTOR
3. RANDOM ACCESS MEMORY (RAM)
4. INSTRUCTION REGISTER (IR)
5. CONTROLLER SEQUENCE
6. T-STATES
7. PROGRAM COUNTER (PC)
8. ACCUMULATOR
9. B-REGISTER
10. ADDER/SUBTRACTOR
11. MEMORY (To show Negative sign).
12. OUTPUT REGISTER
13. BINARY TO BCD CONVERTER
14. CLOCK PULSE GENERATOR (IC 555 TIMER)

PROCESS OF DESIGNING EACH COMPONENTS OF SAP-1

PROGRAM COUNTER

A total of 4 inputs viz; CLK', Cp (Counter Enable), Ep (Output Enable) and CLR' are present in a program counter. We will get 4 outputs from that program counter which will be passed into 2 steps:

i. 4 outputs at first connected to the display bar.
ii. Then those outputs are connected to the bus.

DESIGN

As the RAM we were using is 16x8 bit so we could count 0 to 15 using the program counter. That's why we required 4 J-K flip flops to design the asynchronous counter. Then the 4 outputs were transmitted to the PC3, PC2, PC1, PC0 and thus shown in the display bar. After that those outputs were passed through a tristate buffer and after that the 4 outputs w3, w2, w1, w0 were passed into the bus.

Truth Table

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

FIG: Truth table of JK-flip flop

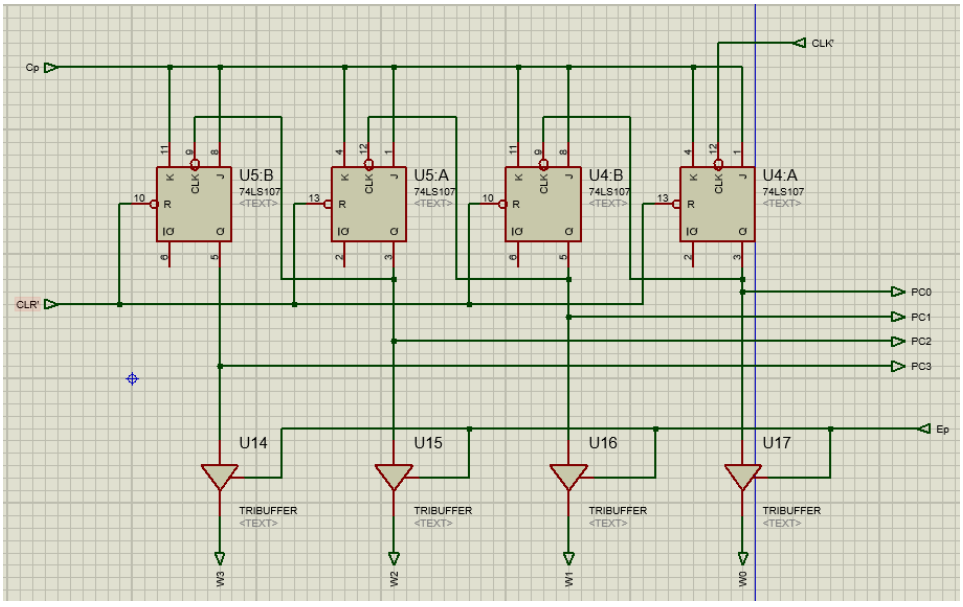


FIG: Circuit diagram of Program Counter

MEMORY ADDRESS REGISTER (MAR)

DESIGN

74LS173 IC containing 4 D-flip flops was used as MAR as it was holding the 4-bit address and transferring it to the RAM through address selector. Lm' input was connected to the input enables (E1, E2) which are active low. So, the value 0 of Lm' caused MAR to store the 4-bit address. CLR (Connected to MR) and CLK (Connected to CLK) did what they usually do. Output enables (OE1', OE2') was grounded to keep output active as we did not need to control the output.

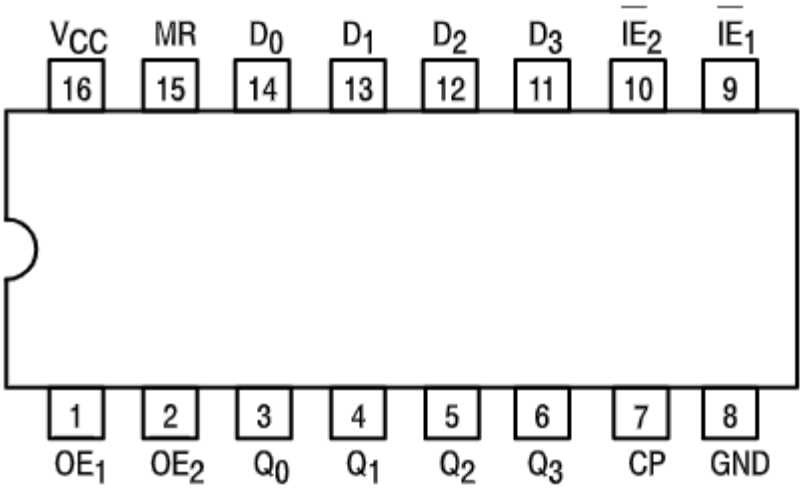


FIG: 74LS173 IC

PIN NAMES

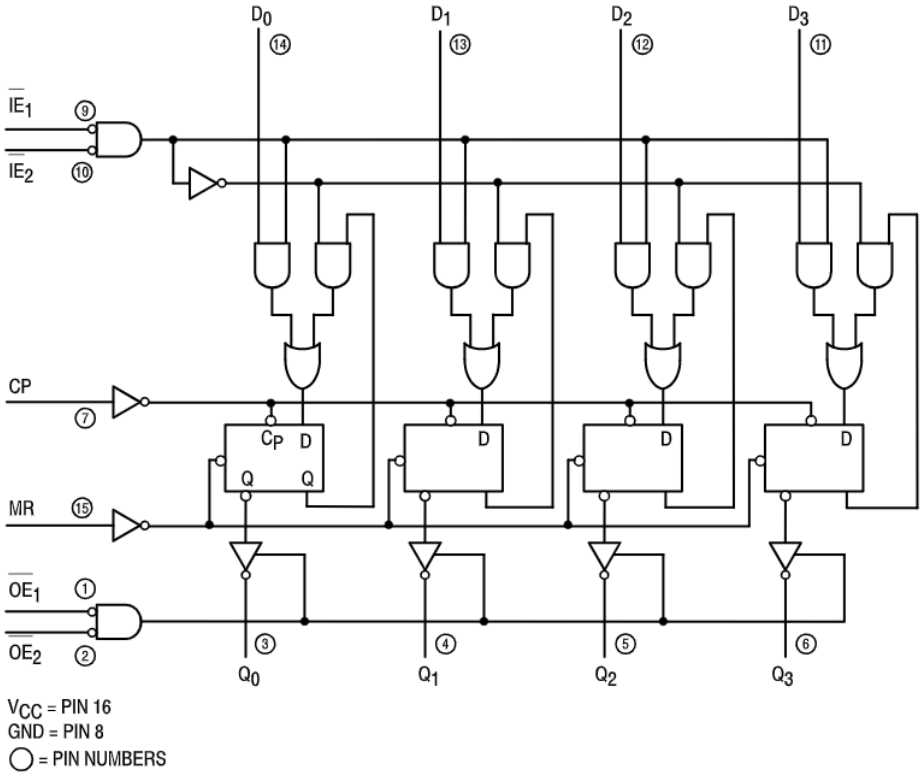
D_0-D_3	Data Inputs
$\overline{IE}_1-\overline{IE}_2$	Input Enable (Active LOW)
OE_1-OE_2	Output Enable (Active LOW) Inputs
CP	Clock Pulse (Active HIGH Going Edge) Input
MR	Master Reset Input (Active HIGH)
Q_0-Q_3	Outputs (Note b)

TRUTH TABLE

MR	CP	$\overline{IE_1}$	$\overline{IE_2}$	D_n	Q_n
H	x	x	x	x	L
L	L	x	x	x	Q_n
L	⌙	H	x	x	Q_n
L	⌙	x	H	x	Q_n
L	⌙	L	L	L	L
L	⌙	L	L	H	H

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

LOGIC DIAGRAM



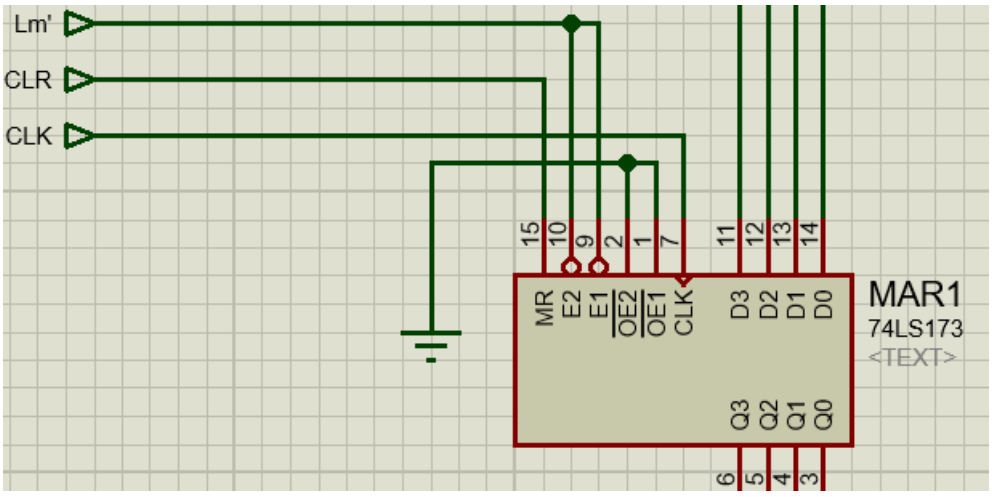


FIG: Circuit diagram of MAR

ADDRESS SELECTOR

The main function of address selector is to determine whether to take the address from MAR or from manual input and then pass that information to the RAM so that it can do its work accordingly. To chose between MAR and manual input PROG(0)/RUN(1) is added. 0 implies manual input and 1 implies to take address from MAR.

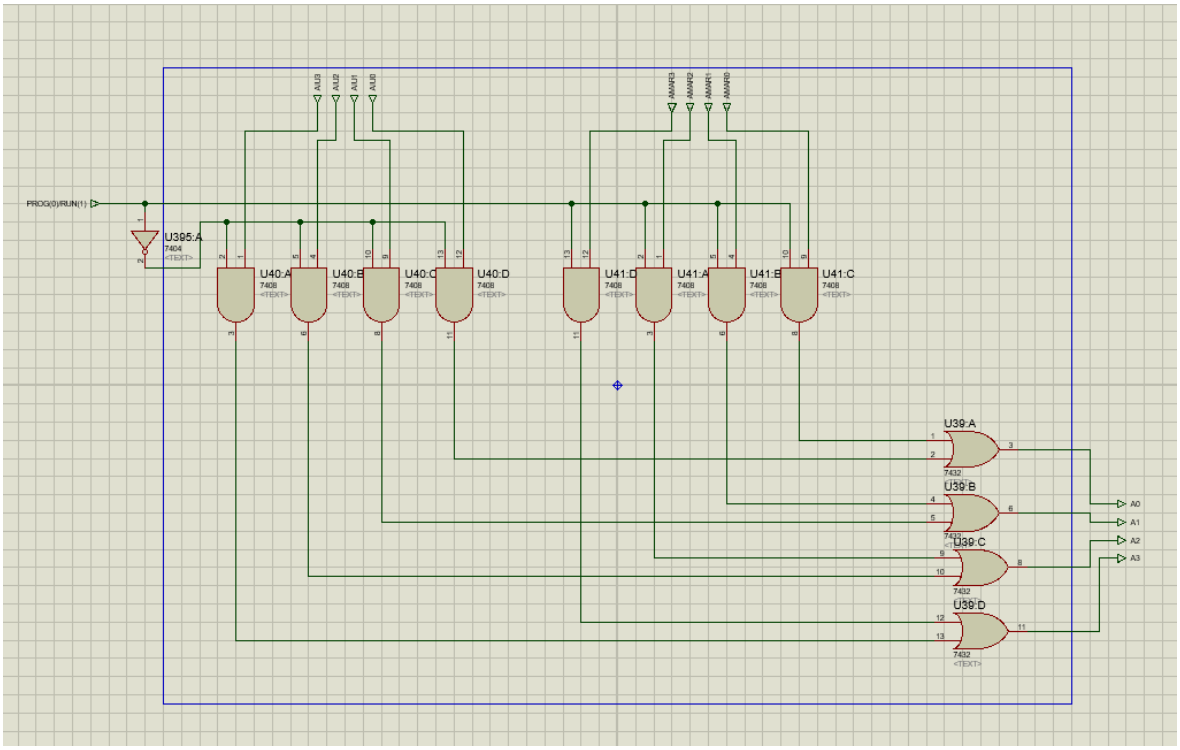


FIG: Circuit diagram of Address Selector

RANDOM ACCESS MEMORY(RAM)

DESIGN

A 16×8 bit ram has been designed for SAP-1.

A 16x8 bit ram will thus contain:

1) 4 address lines (A0, A1, A2, A3)
2) 8 input lines (I0, I1, I2, I3, I4, I5, I6, I7)
3) 8 output lines (D0, D1, D2, D3, D4, D5, D6, D7)
4) 1 Read and Write input (R(1)/W(0))
5) Output enable (CE')

The main parts of RAM are:

1) 4-to-16-bit decoder
2) Ram cell
3) 4 input or gate (4072)
4) Buffer (74LS125)

➢ 4-to-16-bit decoder:

i. 4-to-16-line decoder/demultiplexer IC (74154):

Function Table

Inputs						Outputs														
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = High Level, L = Low Level, X = Don't Care

ii. Not gate (7404)

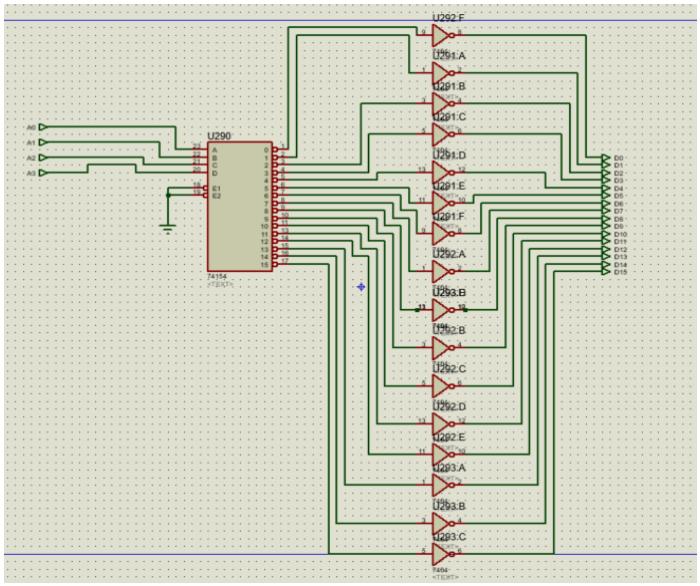


FIG: 4-to-16-line decoder

Truth Table:

E1	E2	A3	A2	A1	A0	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

➢ Ram cell:

Components used for designing the ram cell:

1. Not gate (7404)
2. 3 input And gate (4073)
3. RS latch (Dual type D flip flop)

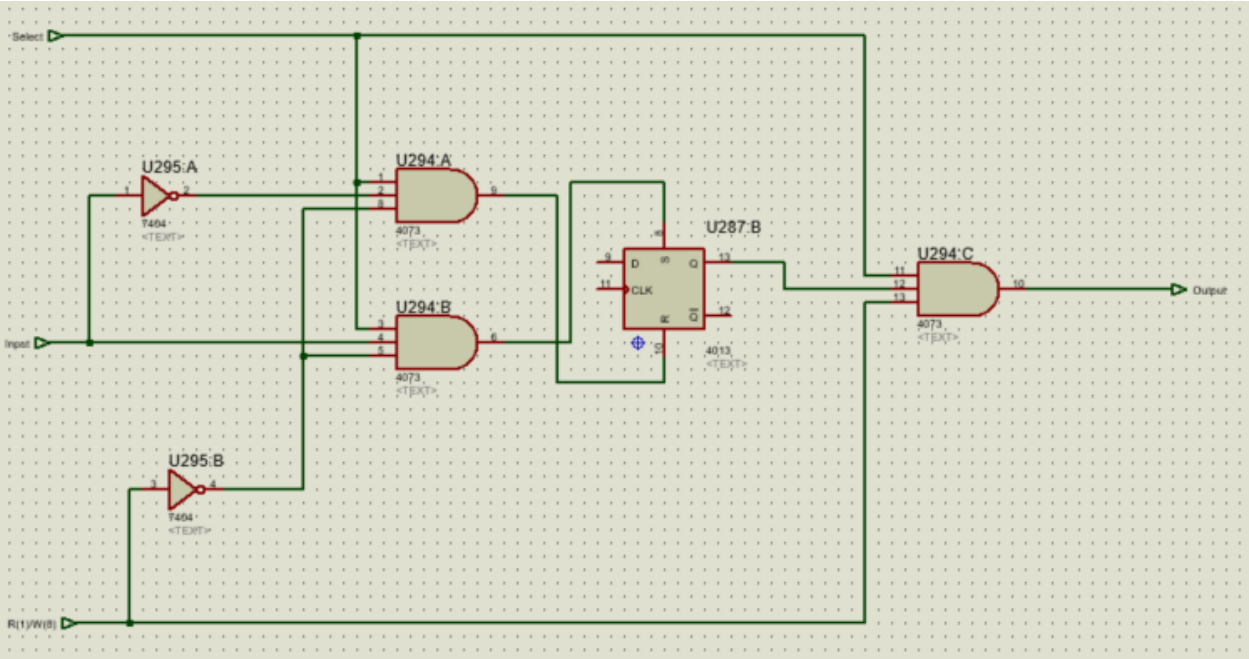


FIG: RAM cell

Truth Table:

Select	R(1)/W(0)	Input	Output
0	x	x	0
1	0	0	0
1	0	1	1
1	1	x	Previously written input

Buffer:

TRUTH TABLES

LS125A

INPUTS		
$\overline{E}$	D	OUTPUT
L	L	L
L	H	H
H	X	(Z)

LS126A

INPUTS		
E	D	OUTPUT
H	L	L
H	H	H
L	X	(Z)

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
(Z) = High Impedance (off)

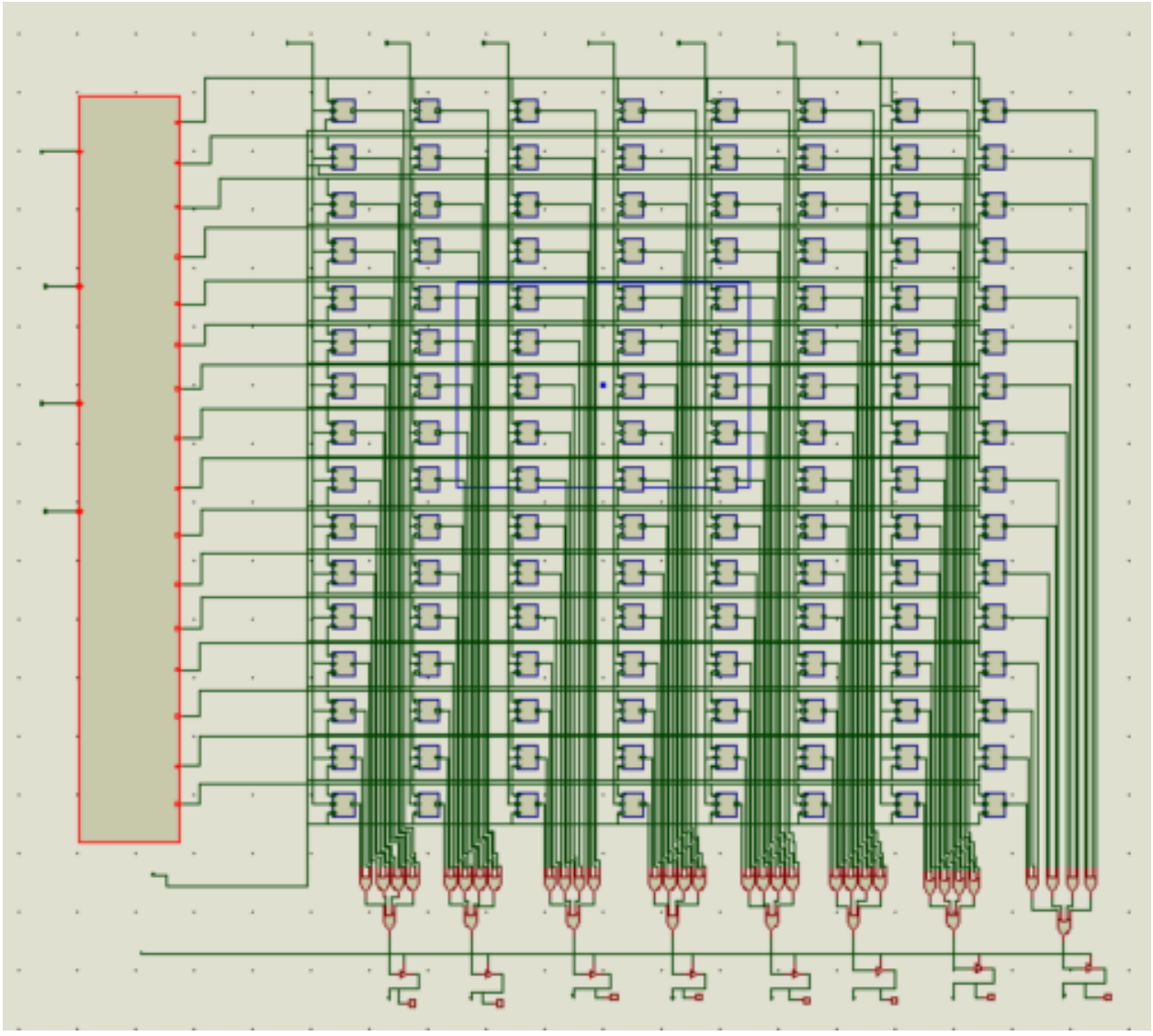


FIG: Circuit diagram of RAM

INSTRUCTION REGISTER(IR)

A total of 8 inputs come to the IR from the bus. Now, the 8 outputs are divided into 2 parts:

i. Upper nibble consisting of the first 4 outputs which are transferred to the controller sequence.
ii. Lower nibble consisting of the last 4 bits which are transmitted to the bus.

DESIGN

Two 74LS173 IC containing 4 D-flip flops each were used to design the IR. The D- flip flop that transferred the 4 bits to the controller sequence had its output enables (OE1', OE2') connected to the ground. Again, the input enables of both the D-flip flops were connected to the Li' input of the IR. Output enables (OE1', OE2') of the D-flip flop that transferred the lower nibble to the bus was connected to the Ei' input. CLK input provided the clockpulse and CLR was used for resetting the values.

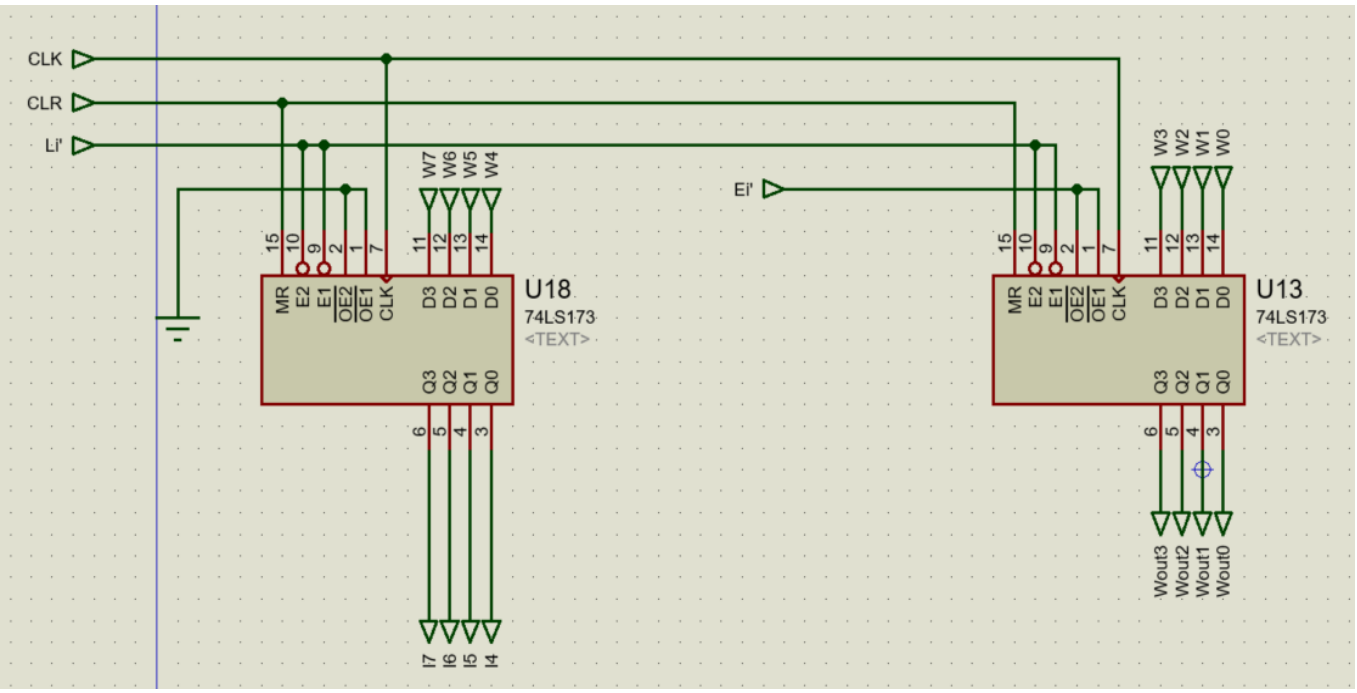


FIG: Circuit diagram of IR

CONTROLLER SEQUENCE

An important component of controller sequence is Ring Counter.

RING COUNTER

DESIGN

6 JK-flip flops were required to construct the ring counter and these flip flops worked for generating the 6 T-states. Ring counter we used was a synchronous counter and the CLR' was connected to Reset of all flip-flops. Outputs of each flip flops were connected to the input of the next flip-flop. And at a particular clockpulse only one T state remained activated.

T1	T2	T3	T4	T5	T6
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

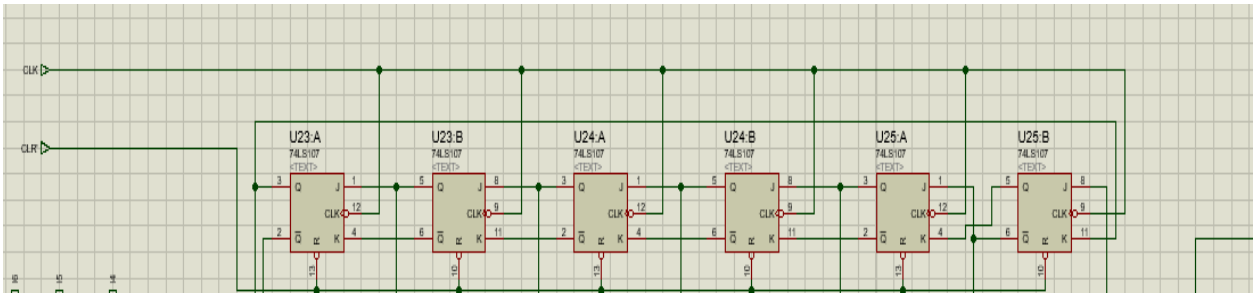


FIG: RING COUNTER

The main function of controller sequence is to design and produce all the control signals viz:

Cp, Ep, Lm', CE', Li', Ei', La', Ea, Eu, Su, Lb', Lo', HLT'

DESIGN

The design of control signals in the controller sequencer was based on 2 factors:

i. The instruction cycles for LDA, ADD, SUB, OUT, HLT.
ii. The 6 T-states for completing each of the 6 microinstructions of each construction cycle.
T1, T2, T3 as fetch cycle.
T4, T5, T6 as execution cycle.

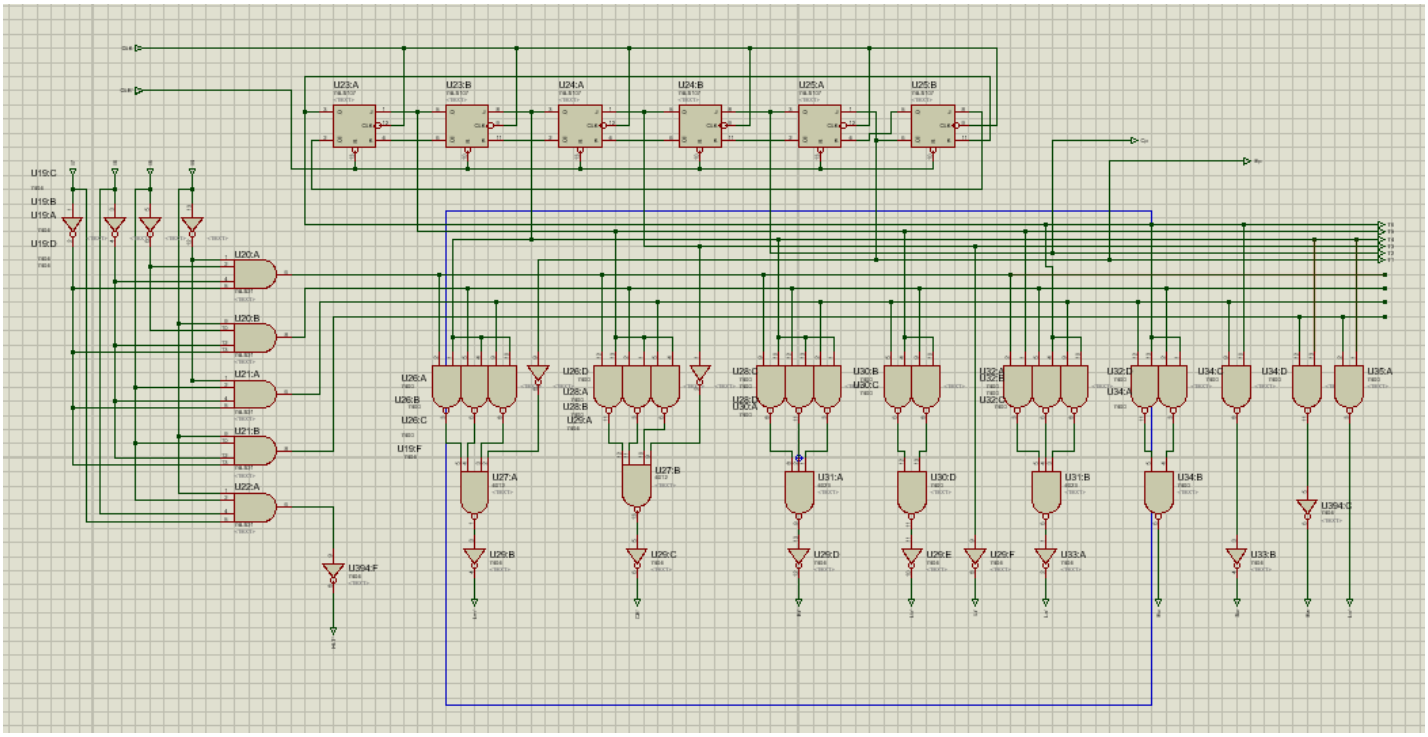


FIG: Circuit diagram of Controller Sequence

T-STATE DESIGN

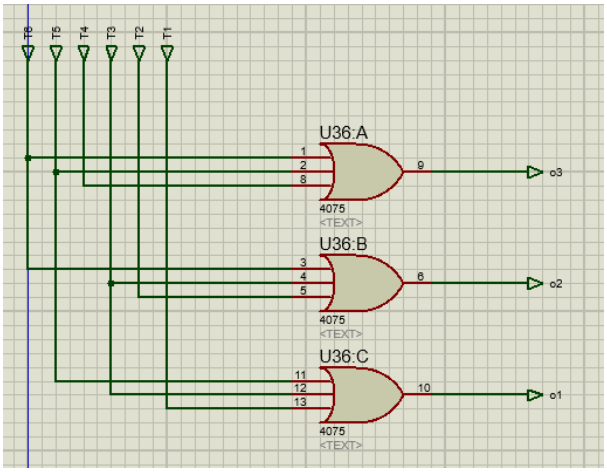


FIG: Circuit diagram of T-state

ACCUMULATOR

At first 8-bit data will be coming from the bus to the accumulator and then stored. Then these stored bits will be transferred to the adder/subtractor for doing the addition/subtraction operation.

CLR, La' (Input Enable), CLK and Ea (Output Enable) are 4 inputs connected to the accumulator. La' activation means the value will be stored in the accumulator from the bus. CLK gives the clockpulse. And the Ea input activation results in transferring data from the accumulator to the bus.

DESIGN

Two 74LS173 IC containing 4 D-flip flops each were used to store the 8 bit data in accumulator. Here, E1, E2 input enables of both the ICs were connected to La'. On the other hand, OE1', OE2' output enables were grounded. MR input was connected to the CLR input and CLK gave the clock pulse. At last, the outputs were connected to the adder/subtractor circuit. For transmitting the 8-bit data to the bus, a buffer was connected to the outputs. Output enable of the buffer was thus connected to the Ea input.

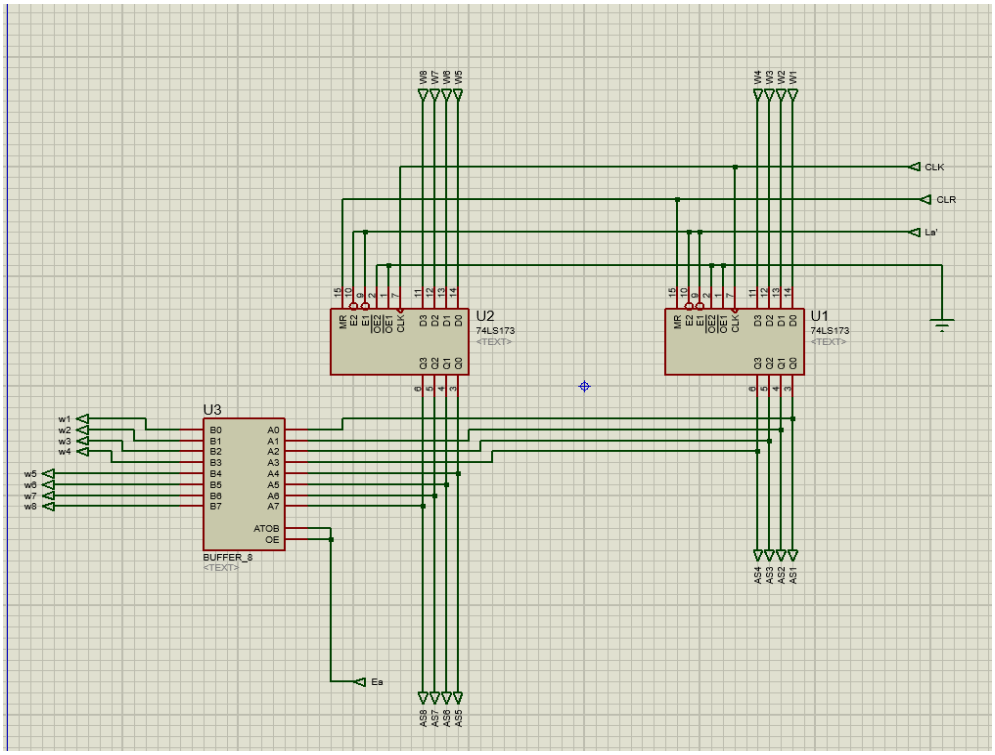


FIG: Circuit diagram of Accumulator

B-REGISTER

A total of 8 inputs will come to the B-register from the bus and after that the output of the register will be sent to the adder/subtractor.

DESIGN

Two 74LS173 IC containing 4 D-flip flops each were used to get 8 inputs (4 from each) for designing the B-register. Output enables were both grounded. Lb' was connected to the input enables and MR was connected to the CLR option. Outputs didn't need to be controlled and was thus transferred directly to the adder/subtractor.

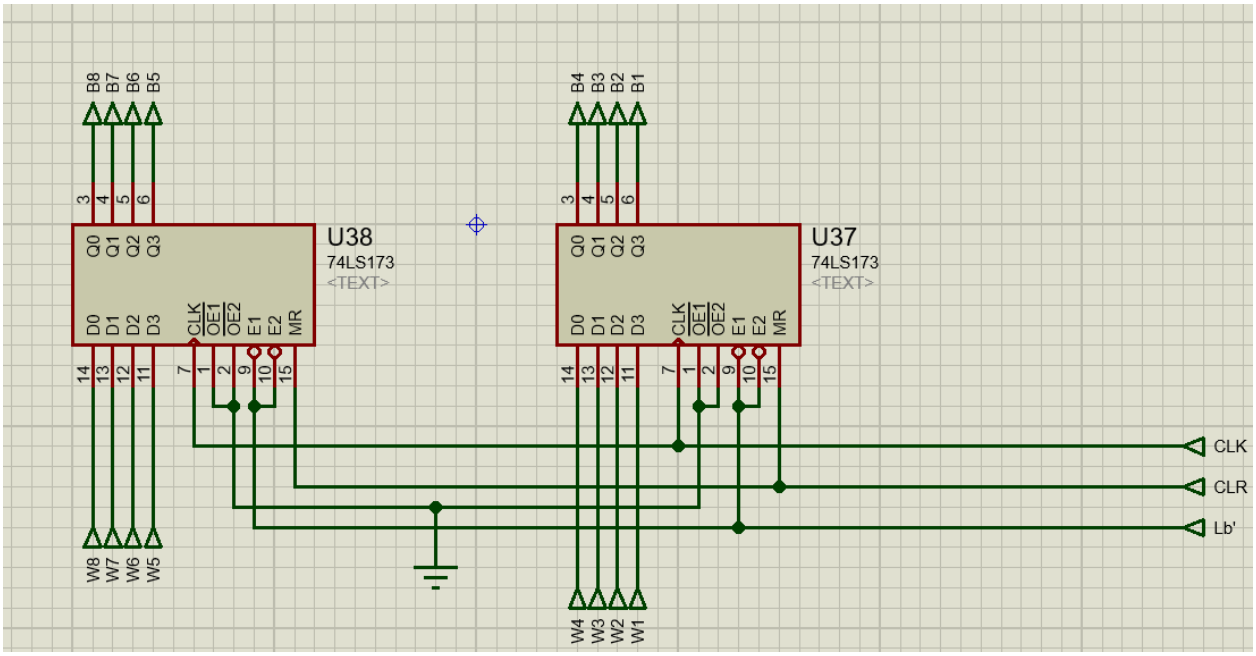


FIG: Circuit diagram of B-REGISTER

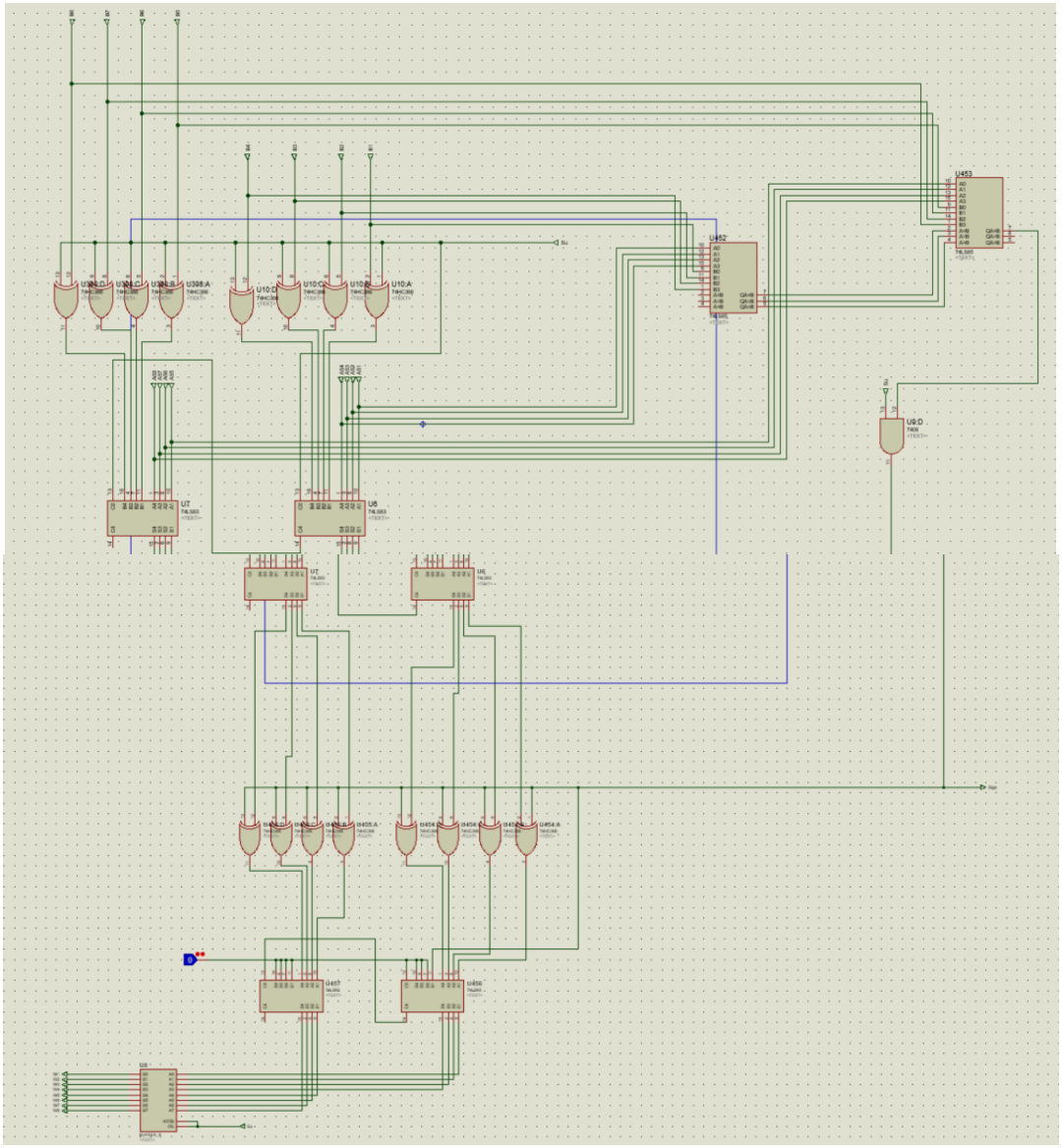
ADDER/SUBTRACTOR (ALU):

Initial design of ALU only had one 8-bit parallel adder which could show addition of two numbers and could only subtract a smaller number from a larger number. But we developed upon it to show negative numbers as output.

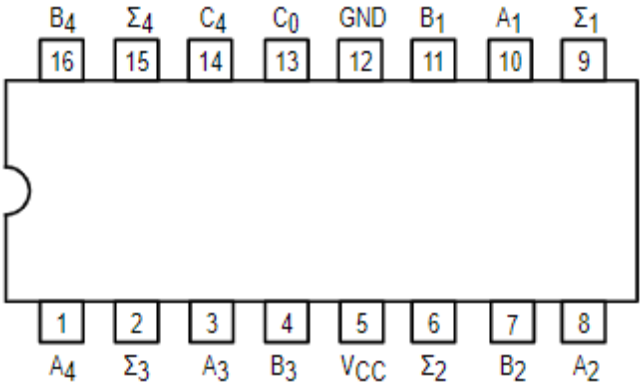
DESIGN

The main components are:

1. XOR gate (74HC386)
2. 4 Bit binary full adder with first carry (74LS83)
3. 4 Bit Magnitude Comparator
4. AND gate (7408)
5. BUFFER



4-bit look ahead carry adder (74LS83):

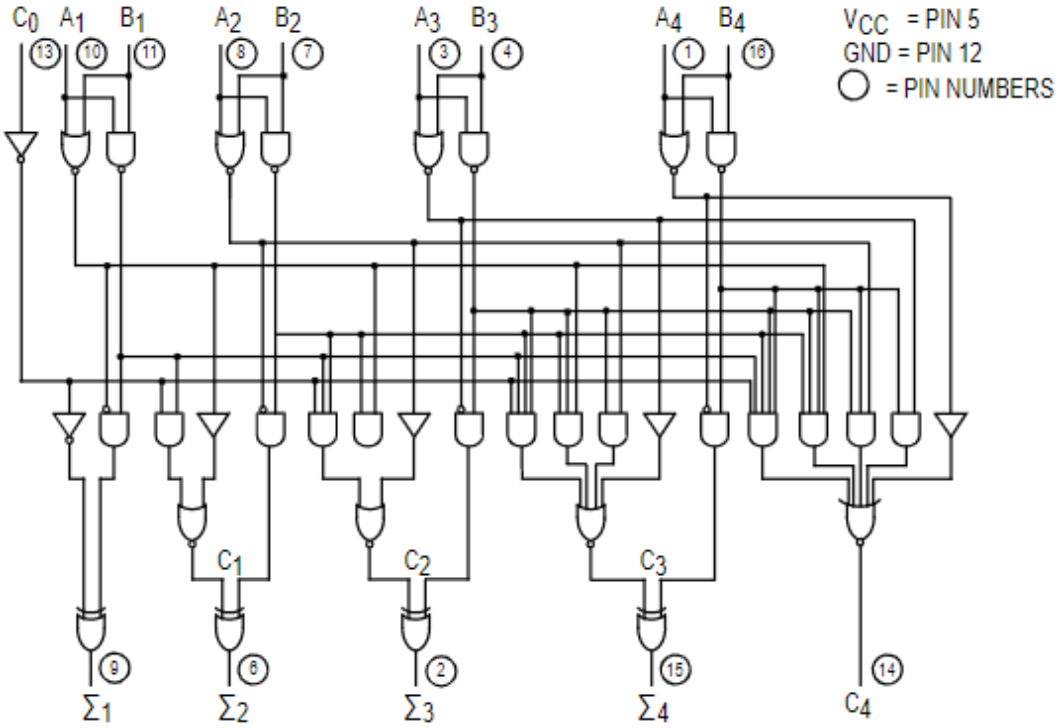


PIN NAMES

$A_1 - A_4$	Operand A Inputs
$B_1 - B_4$	Operand B Inputs
C_0	Carry Input
$\Sigma_1 - \Sigma_4$	Sum Outputs (Note b)
C_4	Carry Output (Note b)

Temperature Ranges.

LOGIC DIAGRAM



This adder subtractor circuit performs 2 basic operations. Adding 2 numbers or Subtracting 2 numbers.

We have constructed two 8-bit parallel adder/subtractor each having two 4-bit binary full adder with first carry (74LS83) and 8 XOR gates.

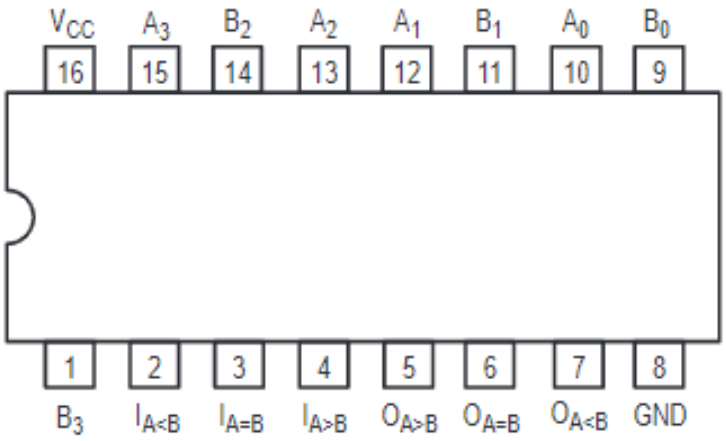
For adding operation:

Input comes from Accumulator (AS1, AS2, AS3, AS4, AS5, AS6, AS7, AS8) and B register (B1, B2, B3, B4, B5, B6, B7, B8)

In case of addition, it doesn't matter which input is larger.

When Su input is 0 it implies addition operation. In this case only the first 8-bit parallel adder is active. But the second one is inactive and just acts like a buffer and lets the output form the first adder circuit to pass.

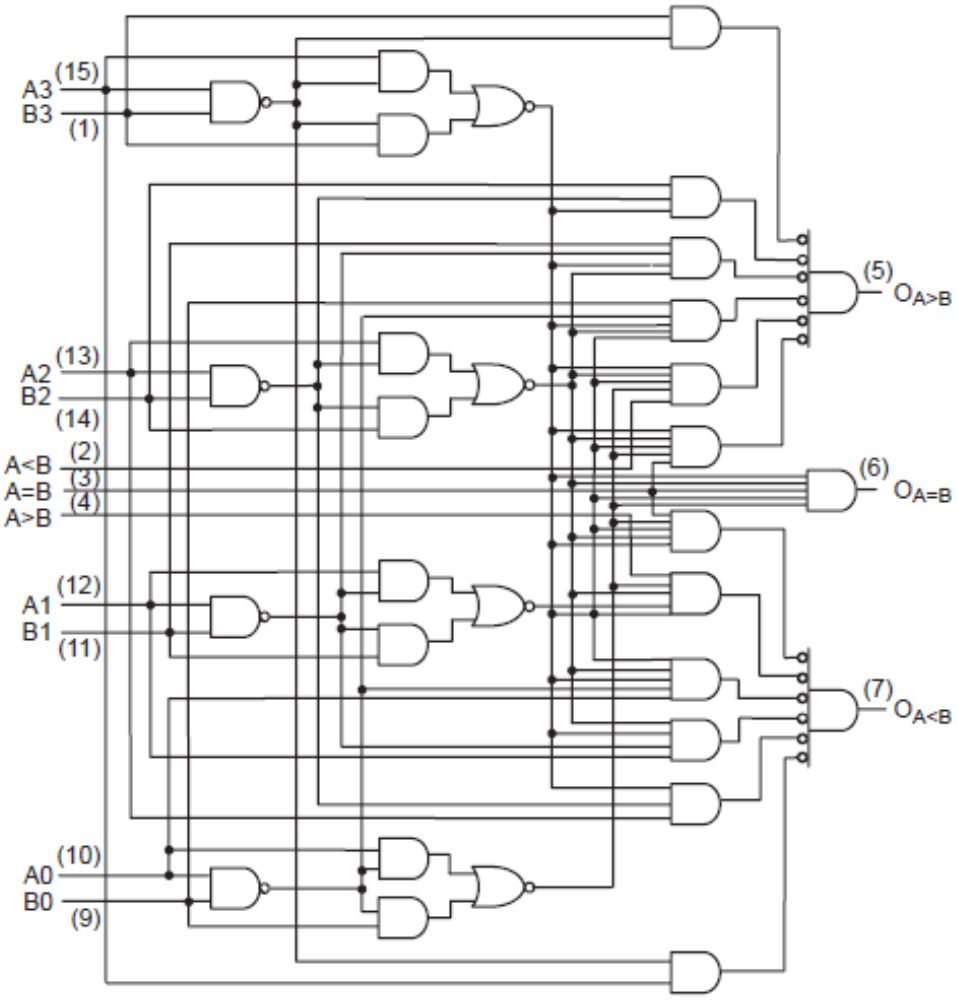
COMPARATOR



PIN NAMES

$A_0 - A_3$, $B_0 - B_3$	Parallel Inputs
$I_{A=B}$	A = B Expander Inputs
$I_{A<B}$, $I_{A>B}$	A < B, A > B, Expander Inputs
$O_{A>B}$	A Greater than B Output
$O_{A<B}$	B Greater than A Output
$O_{A=B}$	A Equal to B Output

LOGIC DIAGRAM

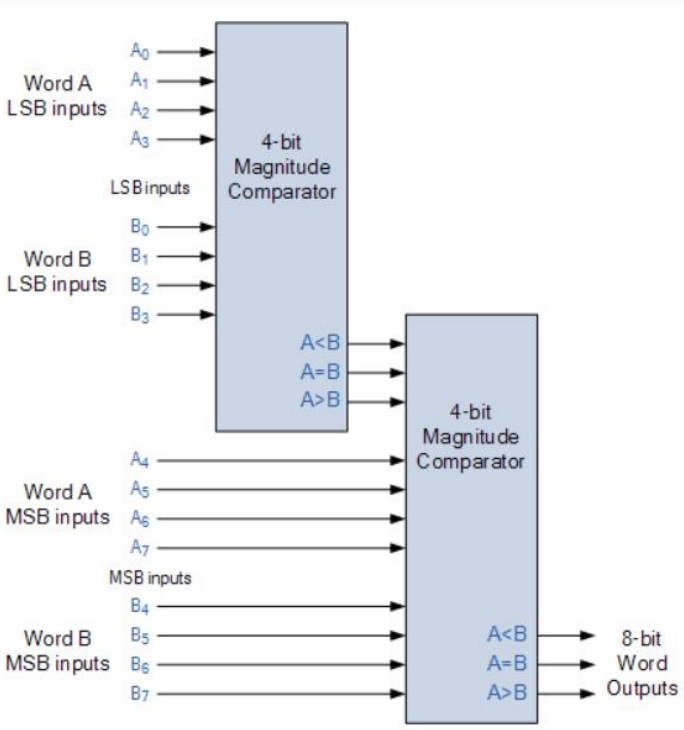


TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A_3,B_3	A_2,B_2	A_1,B_1	A_0,B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	A_1=B1	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L

H = HIGH Level
L = LOW Level
X = IMMATERIAL

8-bit Comparator:



For subtraction operation:

For this case we can have 2 Cases:

Case 1:

A>B which means Accumulator input is larger than B register or A=B which means Accumulator & B register have same value.

When Su is 1 it implies subtraction operation. In this case only first adder circuit will be active.
As the output of AND gate will be zero (Su.QA<B = 1.0=0) the second adder circuit will be inactive and will just act as a buffer and let the output through.

Case 2:

A<B; which means B register input is larger than Accumulator

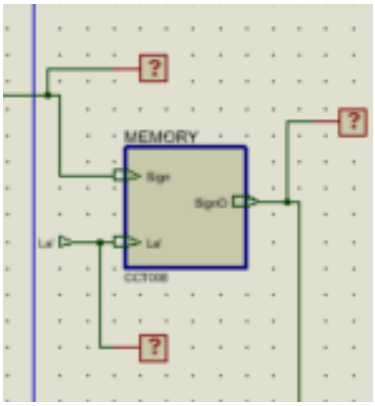
This case was not included in our original design. In our original design this case showed garbage value as output.

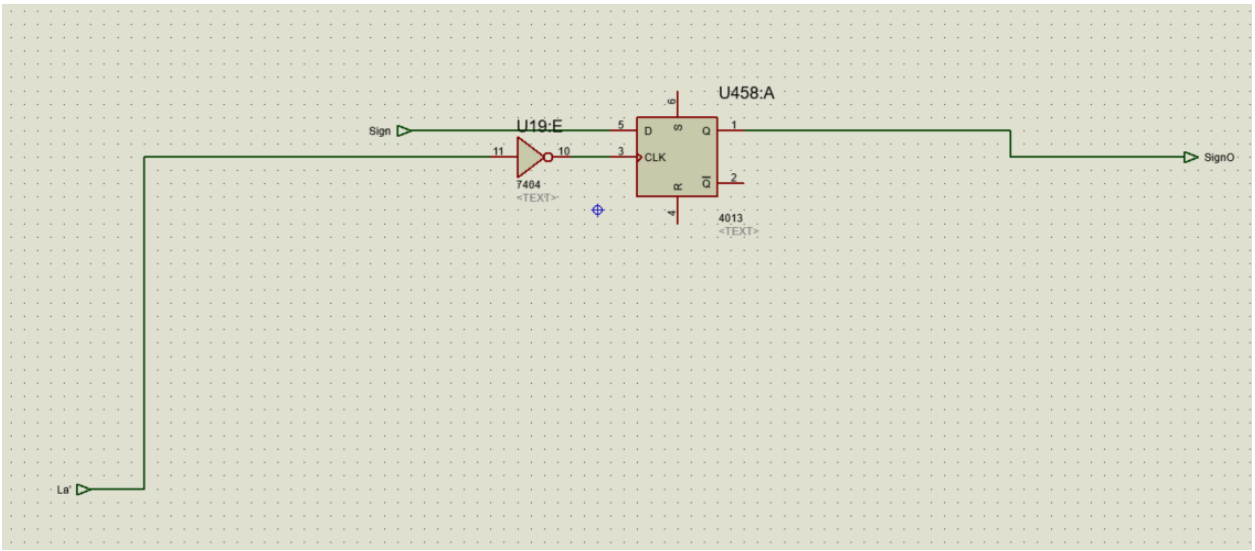
So, in order to have proper output we added another 8-bit parallel adder as in 2's complement method to get negative output we have to perform 2's complement on the original A-B output.

When A<B QA<B of comparator is 1 and also Su is 1. So, AND gate output is 1.

This 1 output goes to the 2nd adder circuit and also goes to the XOR gates.

So, this performs the secondary 2's complement operation. And gives us the correct output.





In order to show negative sign, we used a 7-segment display. As we only want to light up the 'g' part of the display the input should be 1 for 'g' and 0 for all other segments.

Now to generate this sign input we used a memory block. As there are two control signals OUT, HLT after SUB operation. We need to store the output for negative sign.

Memory:

1. D flip flop (4013)
2. NOT gate (7404)

As flip flops are simple components to store a 1-bit data we used a D flip flop.
To trigger this flip flop, we used La'.

Reason of using La' as trigger:

In our 5 instruction sets LDA, ADD, SUB, OUT, HLT we see La' change from 1 to 0 for only the first 3 instruction sets. For OUT and HLT operation La' remains unchanged. As we needed to trigger our flip flop in SUB operation and not to trigger afterwards. We want this data to not change. As La' will not change in OUT and HLT operation so this flip flop will not trigger after SUB operation.

Also, in SUB instruction T state 6 La' changes. So, it will only trigger after the sign input is available.

As La' is active low but our flip flop is positive edge triggered we used a NOT gate to make it active high.

This memory unit has Sign as its input.

Su	QA<B	Sign
0	0	0
0	1	0
1	0	0
1	1	1

So, we see that for only 2nd case of subtraction operation this sign will have 1 value.

OUTPUT REGISTER

The main function of this register is to store the sum/difference and show it to the display option.

DESIGN

Two 74LS173 IC containing 4 D-flip flops each were required to design the output register, Input enables of both the D-flip flops were connected to the Lo'. Output enables were grounded. Activation of Lo' caused the input to be transferred to the register from the bus. Finally, the output was shown in the display option.

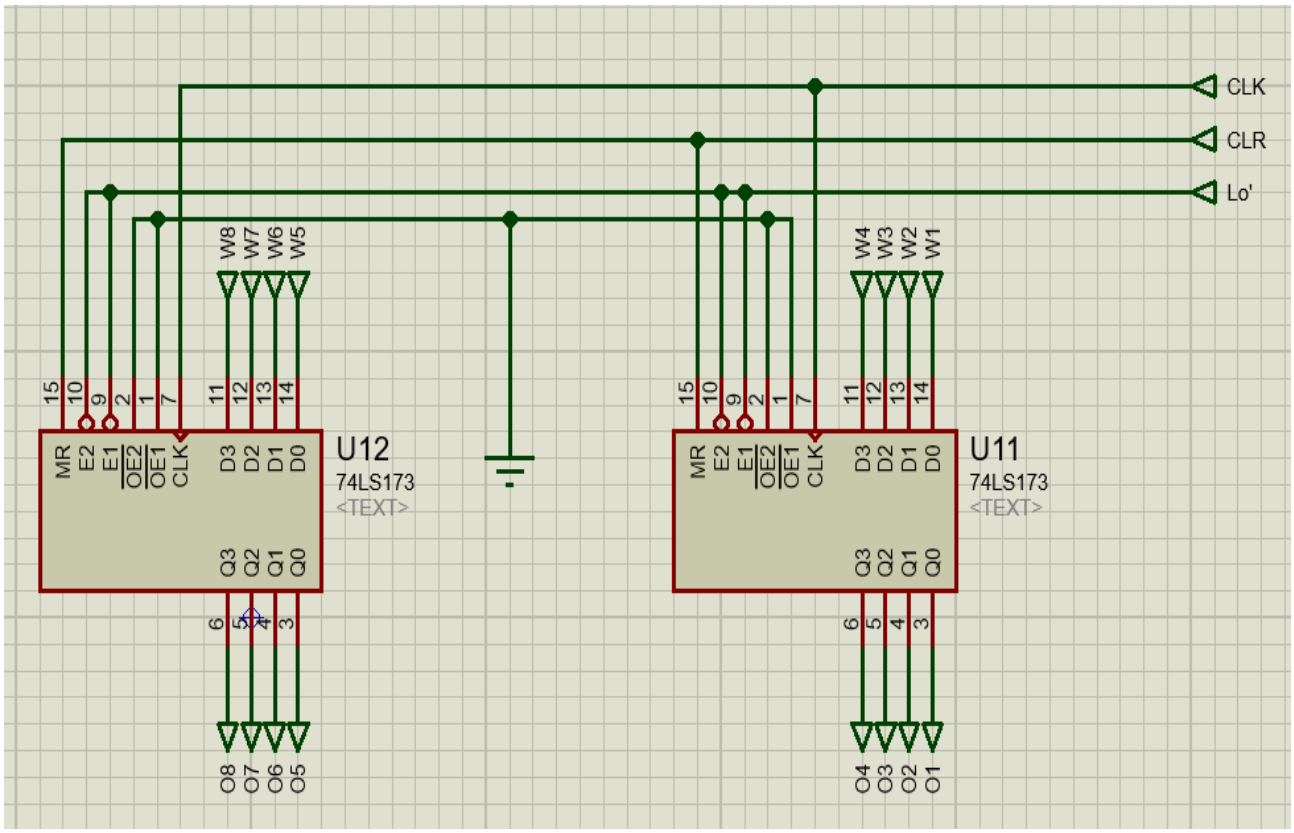


FIG: Circuit diagram of OUTPUT REGISTER

BINARY TO BCD CONVERTER

Output Register provides a 8 bit binary output. To show this binary number displays are needed. But the displays are BCD display. So Binary to BCD converter converts this 8-bit binary to 3 BCD and shows the output in BCD display.

DESIGN

Shift and Add-3 Algorithm

1. Shift the binary number left one bit.
2. If 8 shifts have taken place, the BCD number is in the *Hundreds*, *Tens*, and *Units* column.
3. If the binary value in any of the BCD columns is 5 or greater, add 3 to that value in that BCD column.
4. Go to 1.

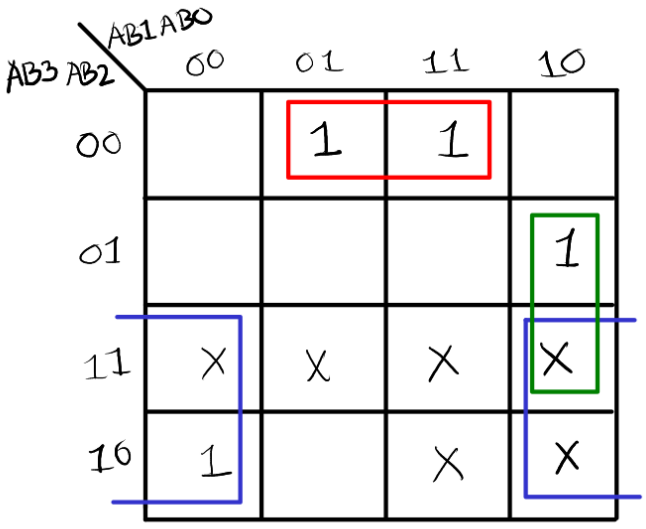
Source: https://johnloomis.org/ece314/notes/devices/binary_to_BCD/bin_to_bcd.html

TRUTH TABLE for Shift Add 3 block

$A[3]$	$A[2]$	$A[1]$	$A[0]$	$S[3]$	$S[2]$	$S[1]$	$S[0]$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

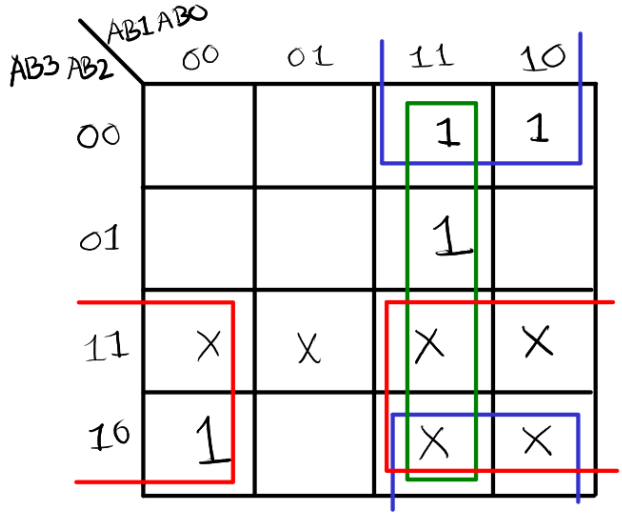
K-MAP SIMPLIFICATION

K-MAP FOR S3



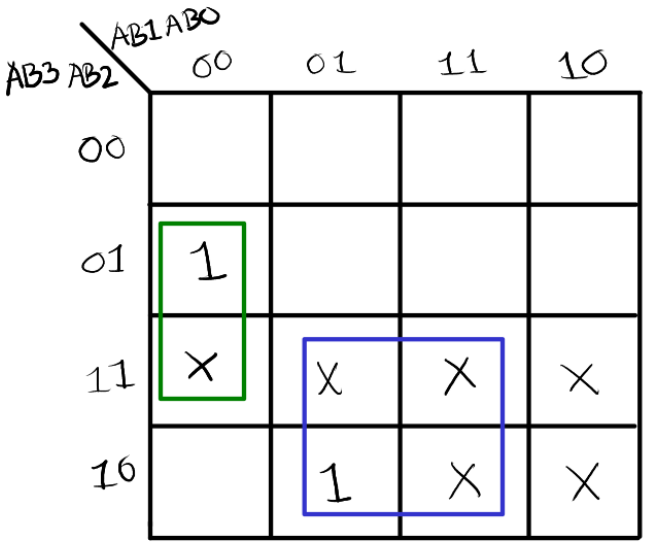
S3=AB3AB0'+ AB3'AB2'AB0+AB2AB1AB0'

K-MAP FOR S2



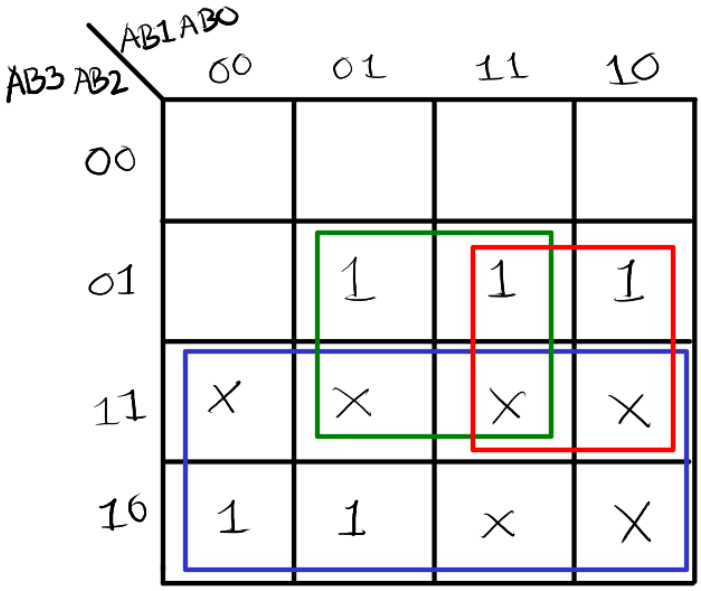
S2 = AB2'AB1+AB1AB0+AB3AB0'

K-MAP FOR S1



S1= AB3AB0 + AB2AB1'AB0'

K-MAP FOR S0



S0=AB3+AB2AB0+AB2AB1

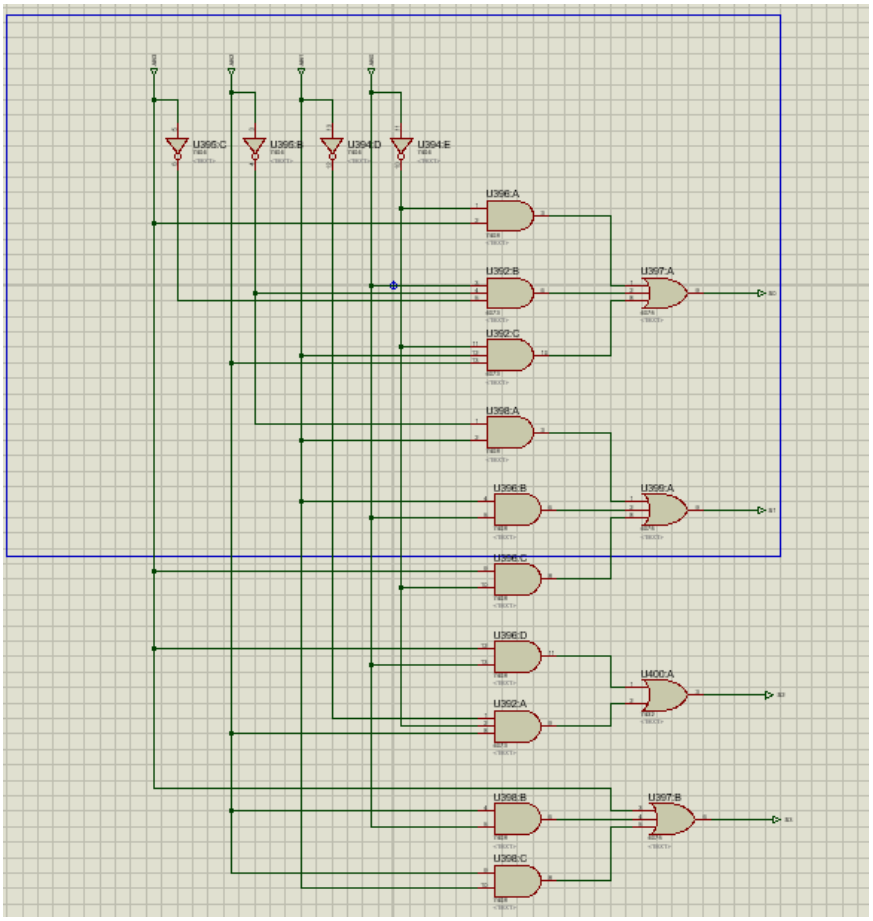


FIG: Circuit diagram of Binary to BCD converter

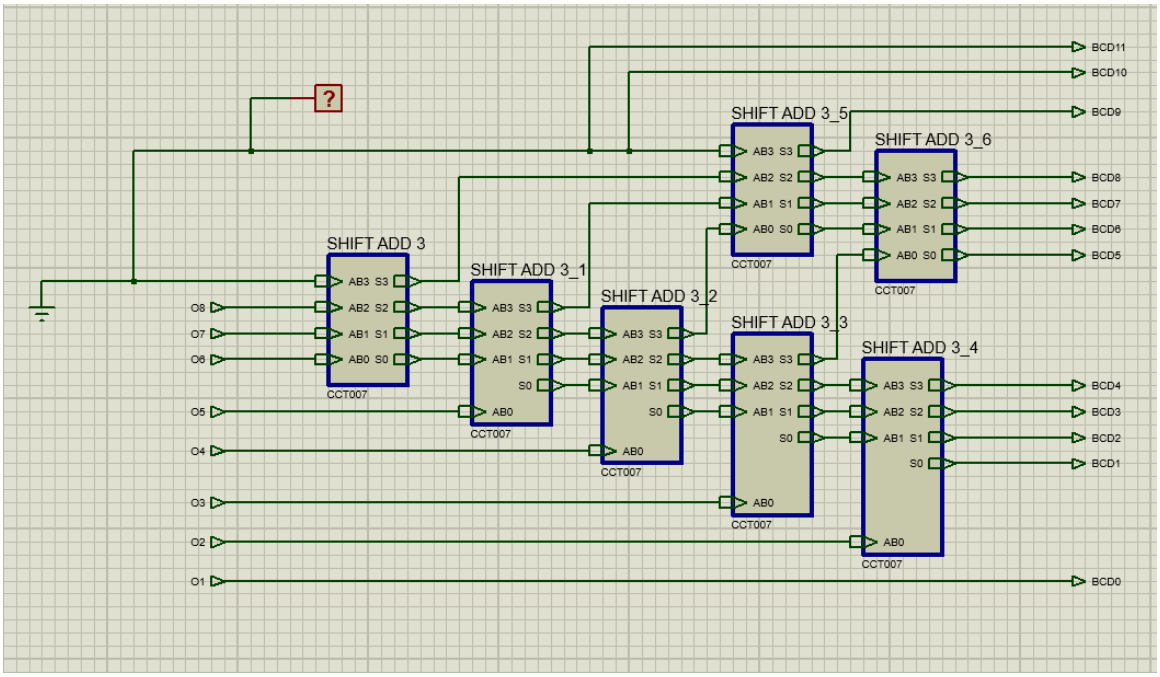
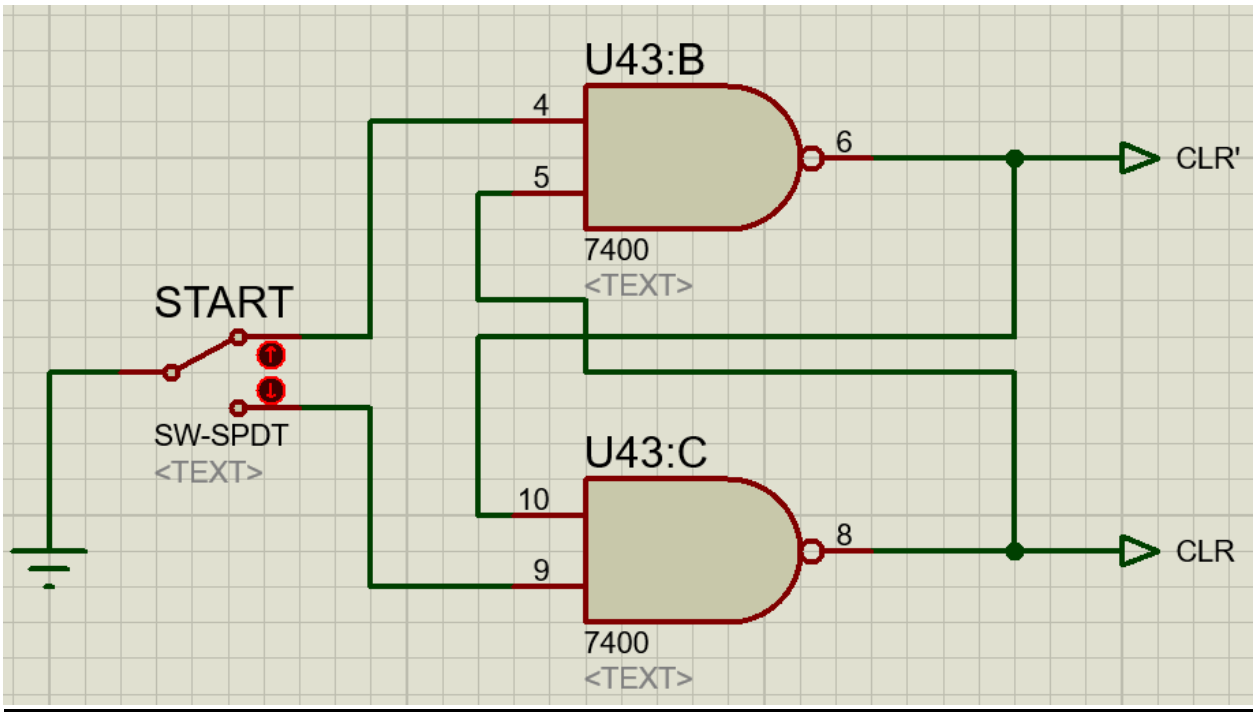


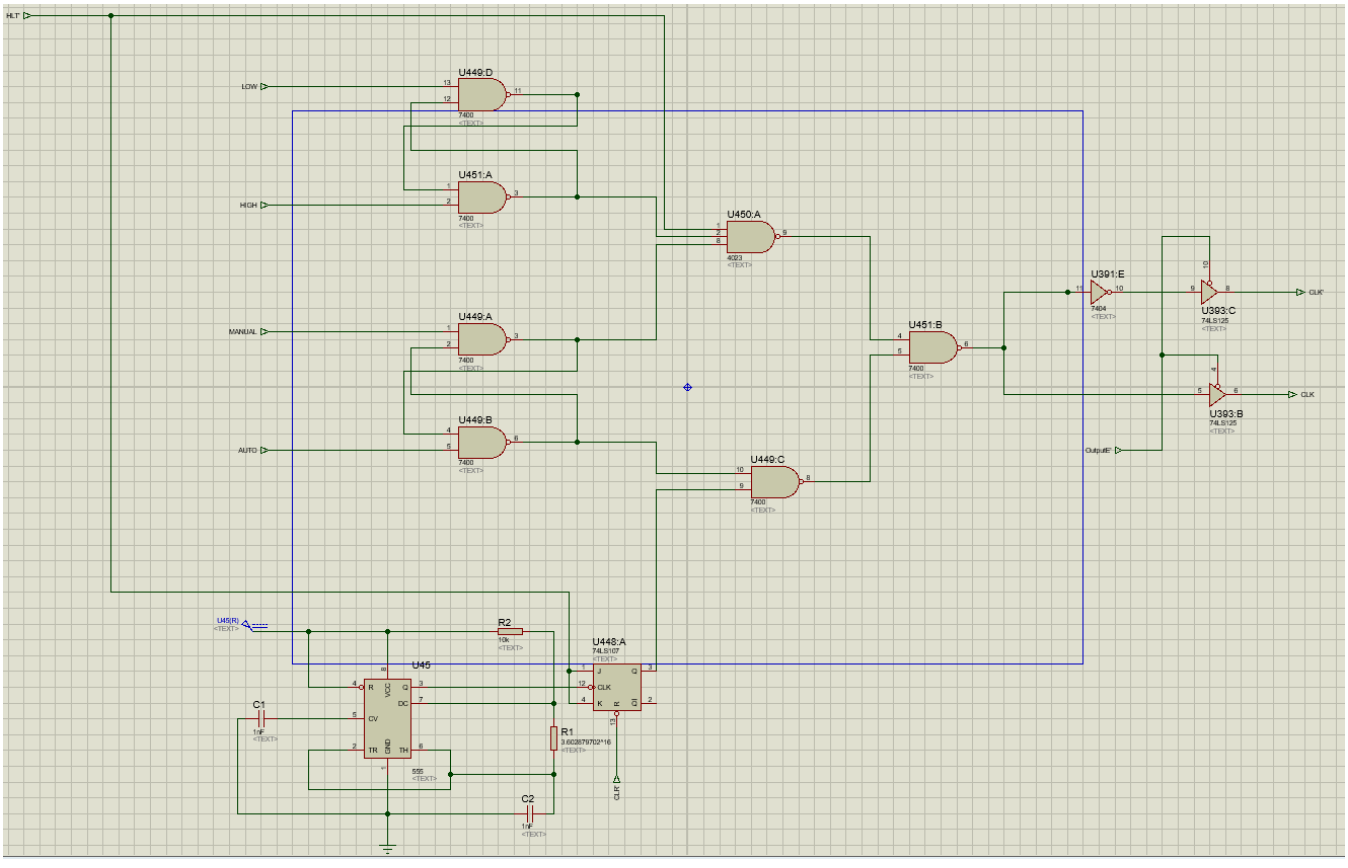
FIG: Block diagram of Binary to BCD converter

CLEAR FUNCTION DESIGN

An S-R NAND latch is used for designing the Clear function.



CLOCK PULSE GENARATOR DESIGN



Problems Faced:

Ram Design:

- ✓ First problem we faced was, after designing the whole RAM circuit and connecting all the components we were not getting the output. After checking again and again we found no problem in the design. After searching the internet, we found that every component in every subcircuit should have unique name.
  To solve this problem, we had to go to each child sheet, and execute global annotator from tool for current branch or sheet with mode set to total. This ensured unique name for every component and thus our RAM was working perfectly.

- ✓ Another problem or mistake was that we did not connect the buffer enable pin to correct input. For this our RAM was not giving output when it should have rather it was outputting when it was not supposed to.

Bus:

- ✓ We noticed that two components RAM and Output register were outputting to the bus at the same time. This was because of the reason mentioned previously. After solving the RAM output enable this problem was solved.

Adder/Subtractor:

- ✓ Initially, the XOR gates we used were not working perfectly but simulation was not showing any error. So, after running several instruction sets we realized that although adder/subtractor was showing outputs but it was incorrect.
  Then we replaced this XOR gate with 74HC386. Now it was working perfectly.

- ✓ Although the modification that was done to show negative value was easy. The main problem that we faced was to show the negative sign. As accumulator value changed for OUT and HLT instruction after SUB instruction. This changed the value of QA<B which in result changed the sign output. In order to solve this, we designed a memory block which solved this problem.

Controller Seq:

- ✓ The first problem we faced was wrong connections. For example, CE', Lb', Eu connections were wrong. After troubleshooting this was solved.

- ✓ The biggest problem we faced here was the design of OUT opcode. As we were following the book of Malvino as reference we designed our OUT part to be activated for opcode 1110. But we were testing our SAP-1 for the OUT opcode 0011 as this was specified in the project video and manual. We didn't realize this at first and as a result was not getting the correct output. After releasing the fact, we changed our design.

Clock Generator:

- ✓ This part was very tricky as we had very little knowledge of 555 timer IC but our initial design worked. But when we tried to connect HLT' input to this the simulation was showing error. We added a resistor to output and then it worked. But when we tried to add manual input, it again showed error. So finally, we decided to scrap our initial design and redesigned the whole thing. We initially connected active high buffer to the output. But it showed error. So, we used active low buffer.

DISCUSSION

Designing SAP-1 gave us an idea about the basic structure of a computer. Uses and applications of combinational and sequential logic circuits were also learnt during its design. However, SAP-1 also has many limitations as well. Instruction commands are very limited in SAP-1. As a result, only addition and subtraction operations can be implemented in SAP-1. Multiplication, division and other complex mathematical operations and instruction commands cannot be operated in SAP-1. Still designing this project made us realize how we can design a very basic computer and observe how it does its very basic operations through proteus simulation.