SAP-1 Report

Contents

Project Objective

The objective of this project is to build the Simple As Possible (SAP) – 1 computer in proteus software and optimize it as much as possible. SAP-1 is the first stage in the development towards modern computer. This introduces us to the basic ideas behind microprocessor and computer operation.

Thus, we have constructed our SAP-1 project in proteus completely. We have constructed various modules to make the SAP-1 computer. The details each module is given below.

Clock Pulse Generator

Here, we have used two methods of generating the clock pulse, one is automatic and the other one manual. For automatic operation we have made an Astable multivibrator and for manual operation, Monostable multivibrator using the 555 timer IC. For choosing between the two modes, a bistable multivibrator has been used. Then with combinational circuit the whole pulse generator has been completed. Details are as follows:

Astable Multivibrator

Astable multivibrator is one which have no stable state, i.e., it'll have two states HIGH and LOW running between each other. The internal theoretical circuit and proteus implementation of it is as follows:

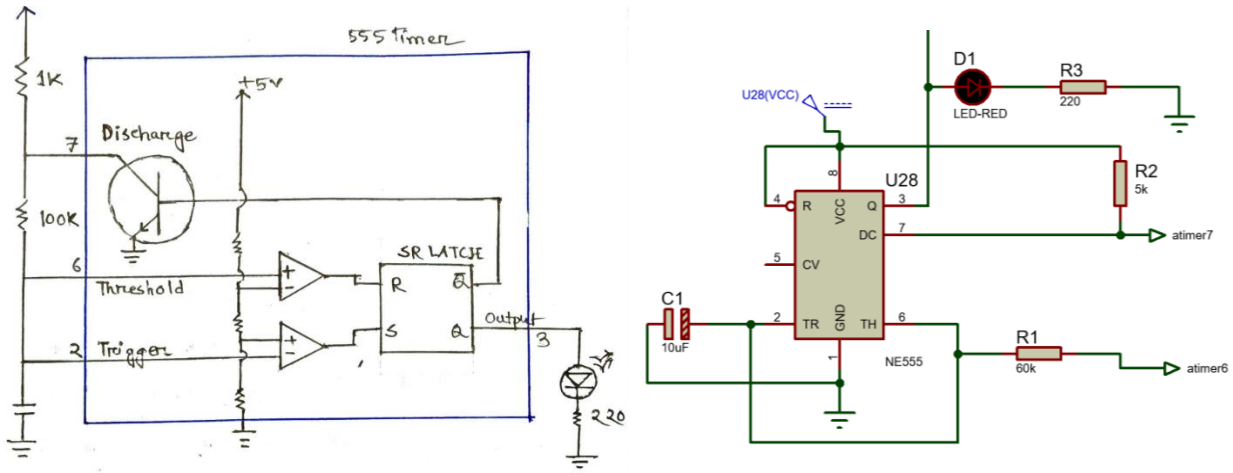


Fig: Astable Multivibrator using 555 timer IC

There is a potentiometer between the output pin atimer7 and atimer6 which is connected outside the subcircuit. This is to control the frequency of the generated pulse.

Monostable Multivibrator

The monostable multivibrator has one stable state which is when we press the button it'll toggle to high for certain amount of time. Here we could've used just a switch between power source and output. But this creates a problem named bouncing. To avoid this, we used the monostable multivibrator.

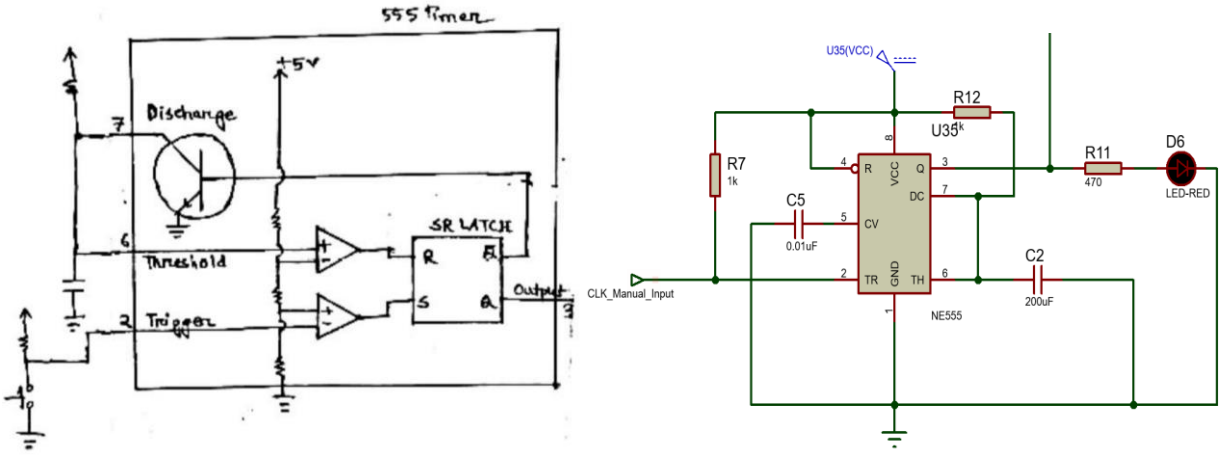


Fig: Monostable Multivibrator using 555 timer IC

Bistable Multivibrator

Again, to choose between the monostable and the astable multivibrator we could've used a switch but since it'll creating bouncing problem, we used the bistable multivibrator as a switch. The bistable multivibrator switches between High and Low. So, for High automatic mode and for Low manual mode will be selected. For this we had to design a combinational circuit.

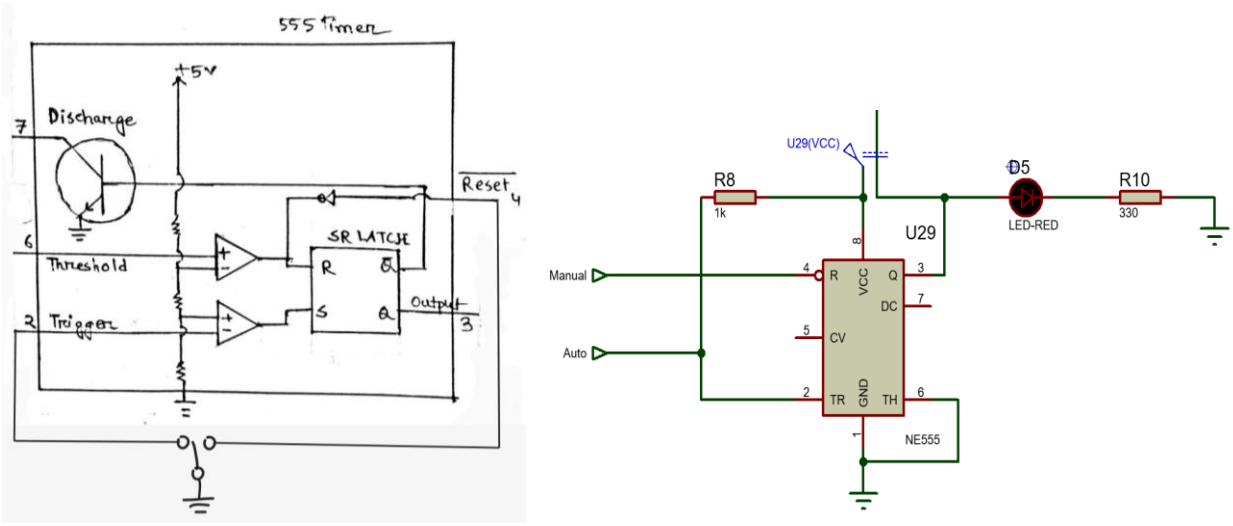
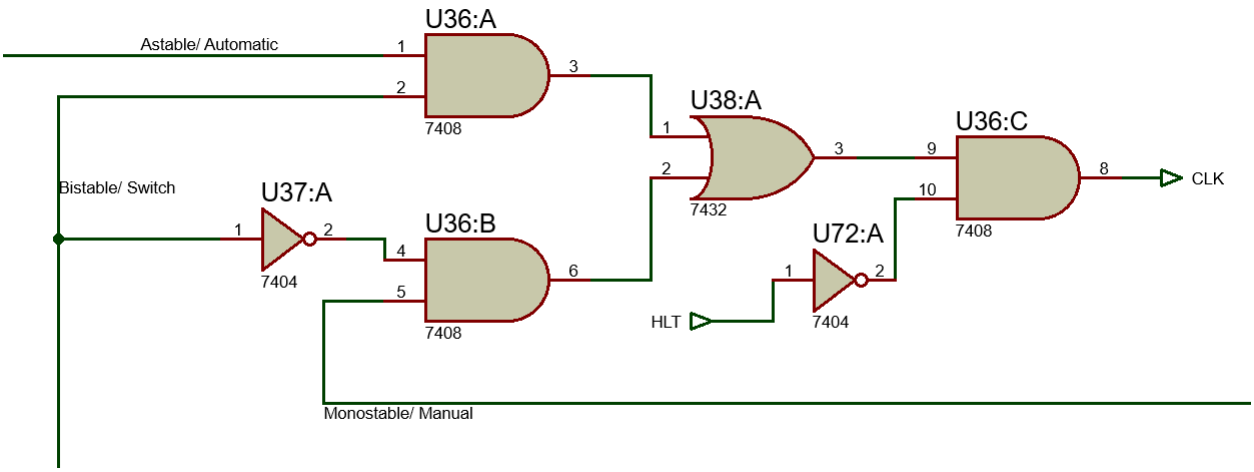


Fig: Bistable Multivibrator using 555 timer IC

Combinational Circuit to Select Automatic or Manual Mode

We made this combinational circuit to toggle between a stable and monostable output.



Truth Table

HLT	bistable	astable	monostable	output
1	x	x	x	0
0	1	0	x	0
0	1	1	x	1
0	0	x	0	0
0	0	x	1	1

Full Clock Circuit

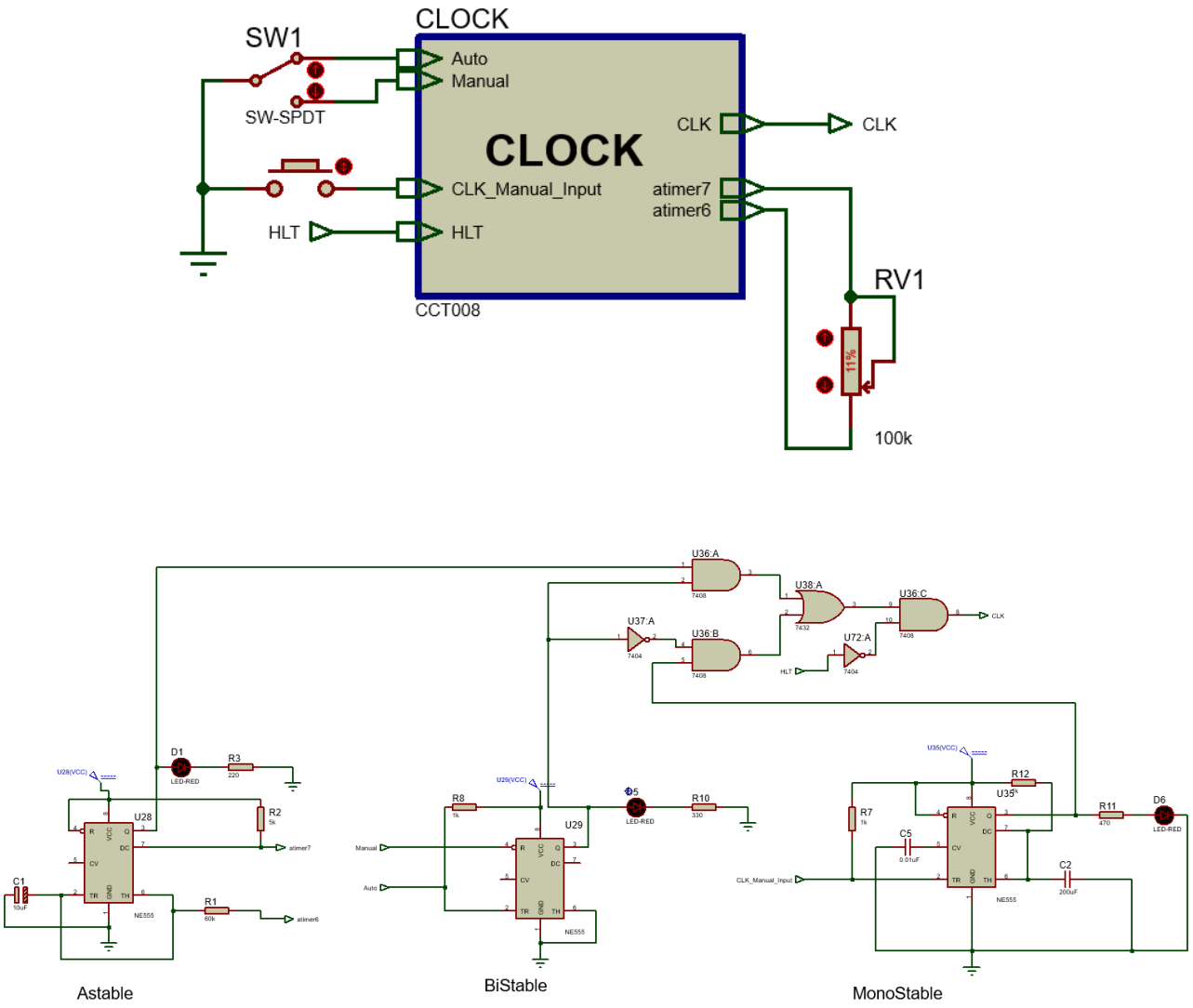


Fig: Full Clock Circuit

Program counter:

The Program counter counts from 0000 (0) up to 1111 (15). It sends counting bits to the RAM so that the instruction saved in the RAM can be fetched. In every clock pulse, its value will be incremented after completion of each instruction cycle (completion of an instruction)

In our case, we used a 4-bit asynchronous up counter. When CE is High counter is enabled which increment the value of the program counter on the next clock pulse. This CE is connected to every J and K input of the JK flip flops.

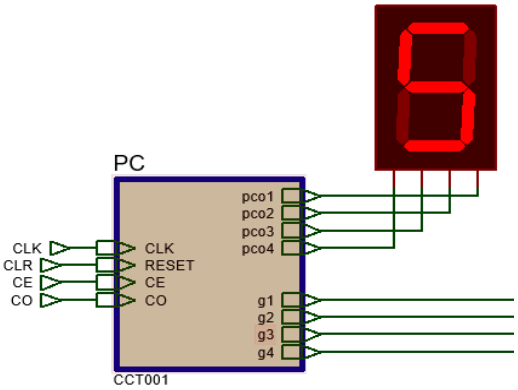


Fig: Program Counter

Circuit Diagram for Program Counter:

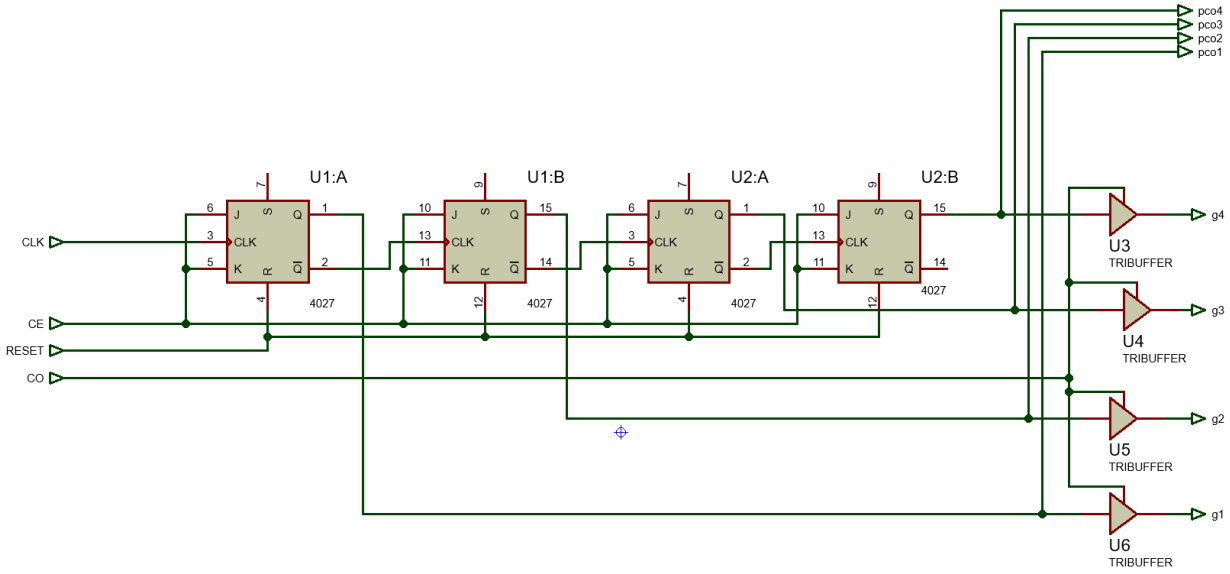
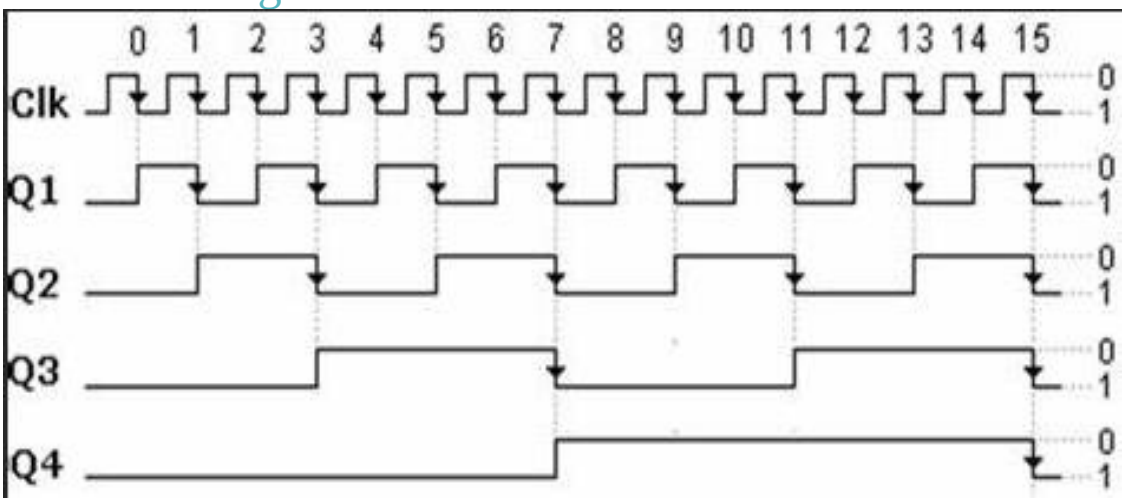


Fig: 4-bit up counter with control Bits

Timeline Diagram:



Input Unit

The input unit is used to provide instruction data to the RAM. Here, address pins are used to select the address of the RAM and Data are stored in the respected address by the Data Inputs. W/R is the read and write pin. It toggles between read and write mode of RAM.

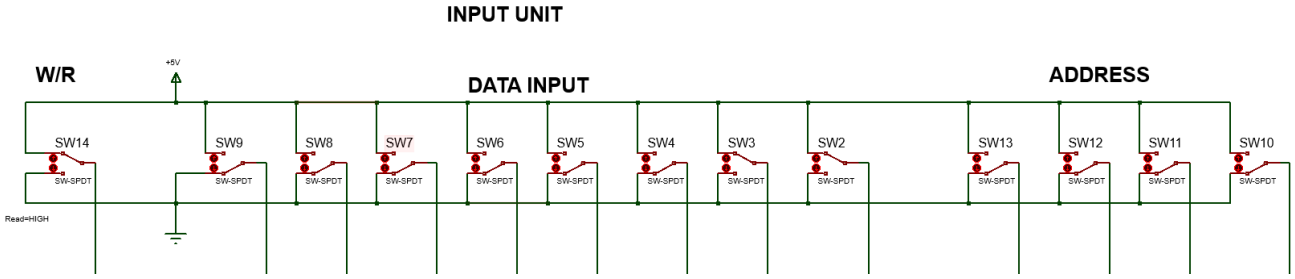


Fig: Input Unit

MAR

Here we have made the MAR with basic Parallel in Parallel out shift register. For the shift register we have used SR- flip flop. Since only four-bit data pass through MAR we have used 4-bit shift register. Now for storing the data in the MAR we used Shift/Load combinational circuit. Here the Shift/Load input is represented by MI instruction. When MI is High, data is shifter to the register and when it is Low data remains stored in the register. MAR sends the address data to the RAM when RUN state is activated.

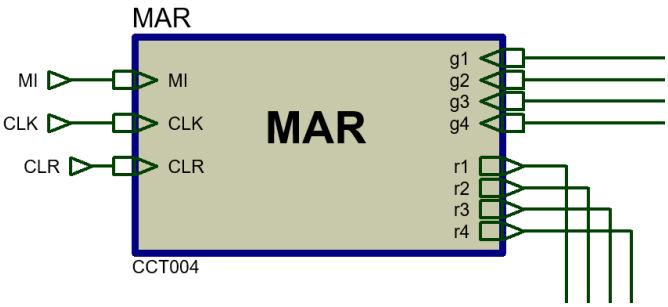


Fig: MAR Module

Truth Table

MI	Input Data	Output
1	1	1
1	0	0
0	x	Memory

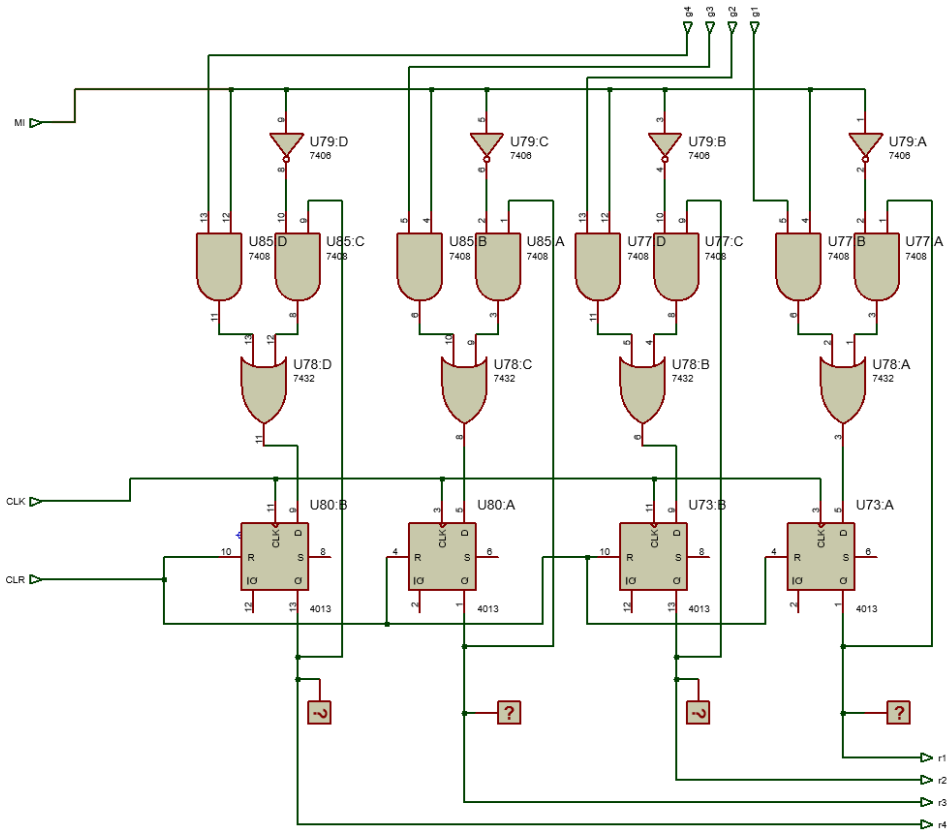
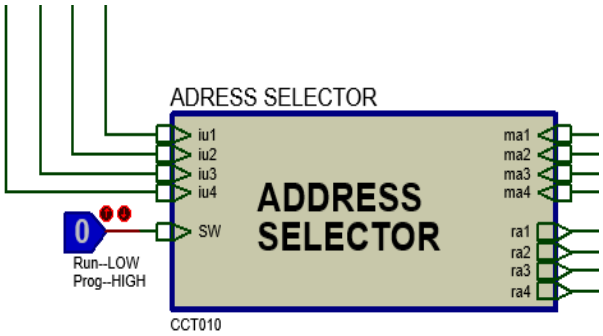


Fig: MAR Circuit

Address Selector

There are two states in SAP, one is program state and another is run state. In the prog state the address is selected manually and in the run state the address is selected automatically or using the information coming from the MAR. If the SW(Prog/Run) input is high the SAP is in Program state and for the case of low SAP is in Run state. This address selector basically works for 4-bit address data. The Address selector's single bit input acts like given truth table



Truth Table

Prog/Run (SW)	Input from IU(iu_n)	Input from MAR(ma_n)	Output (ra_n)
1	1	X	1
1	0	X	0
0	X	1	1
0	X	0	0

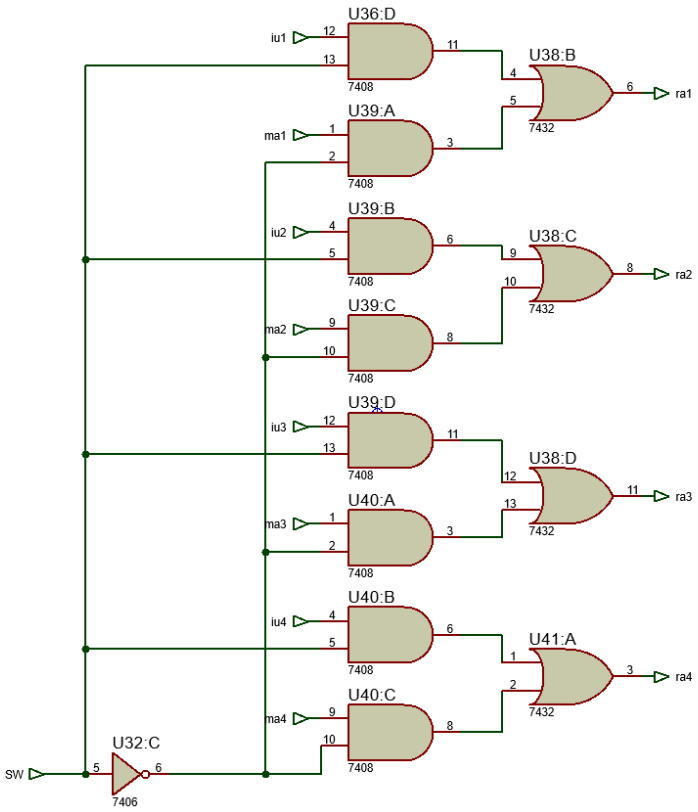
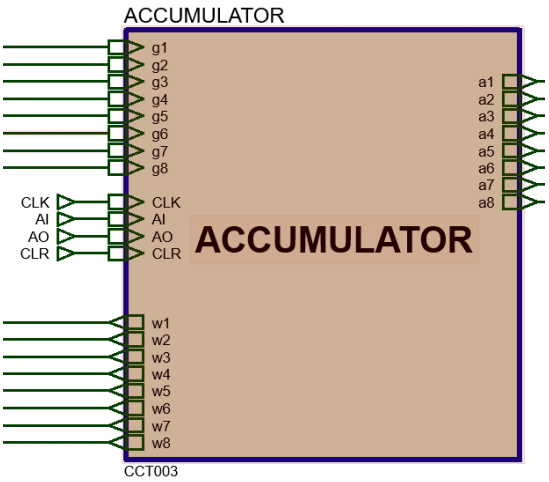


Fig: Address Selector

Accumulator

The accumulator circuit is same as MAR circuit. But the accumulator is an 8-bit parallel in parallel out shift register. And the Shift/Load instruction here is represented as AI instruction. Moreover, an additional instruction AO is used to control the output of the accumulator. For this we added tristate buffer to every output data line. So, when AO is high, data stored are transferred from Accumulator to Bus and when AO is low no data can pass from accumulator. There is also another state of output where no tristate buffers are used. These are sent directly to the ALU.



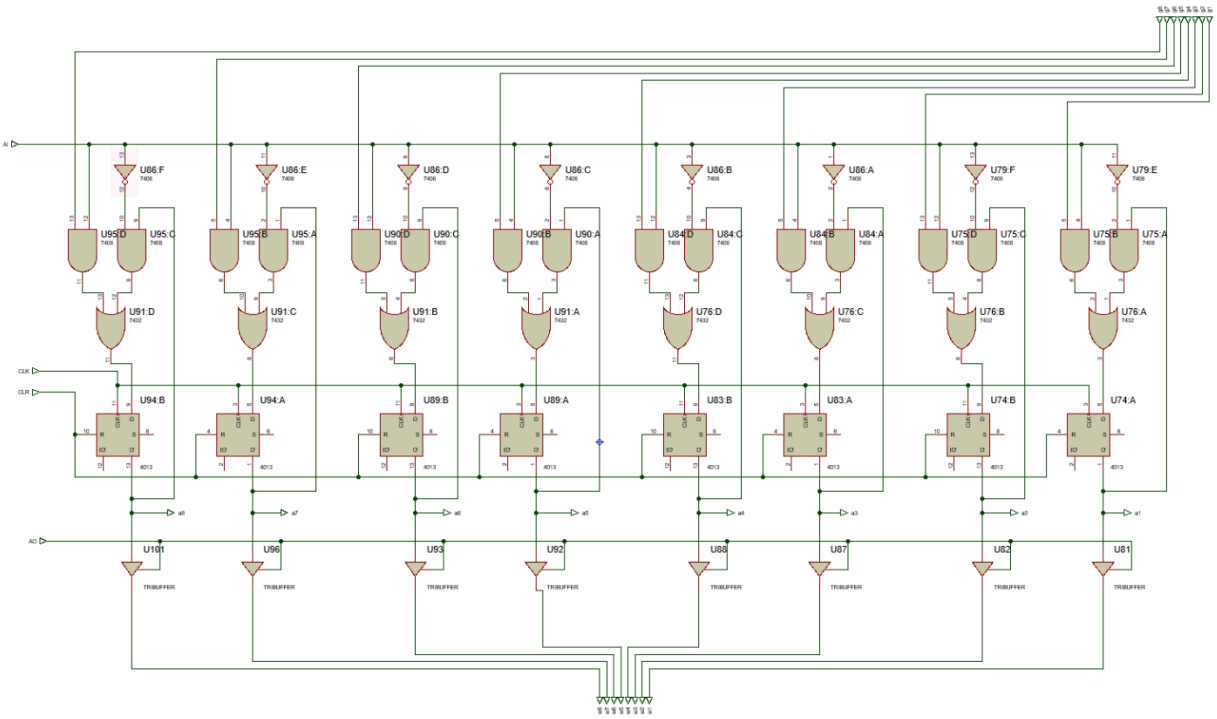
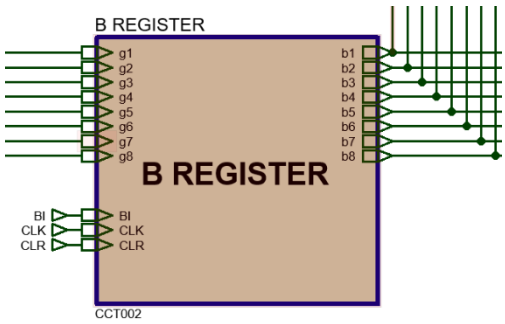


Fig: Accumulator Circuit

B-Register

B- Register is an 8-bit register which is similar to Accumulator only with different instruction. The only instruction used in B register is BI which act as the Shift/Load. Unlike accumulator, there is no output instruction here. So, no tristate buffer has been used and thus the 8-bit data is sent to the ALU.



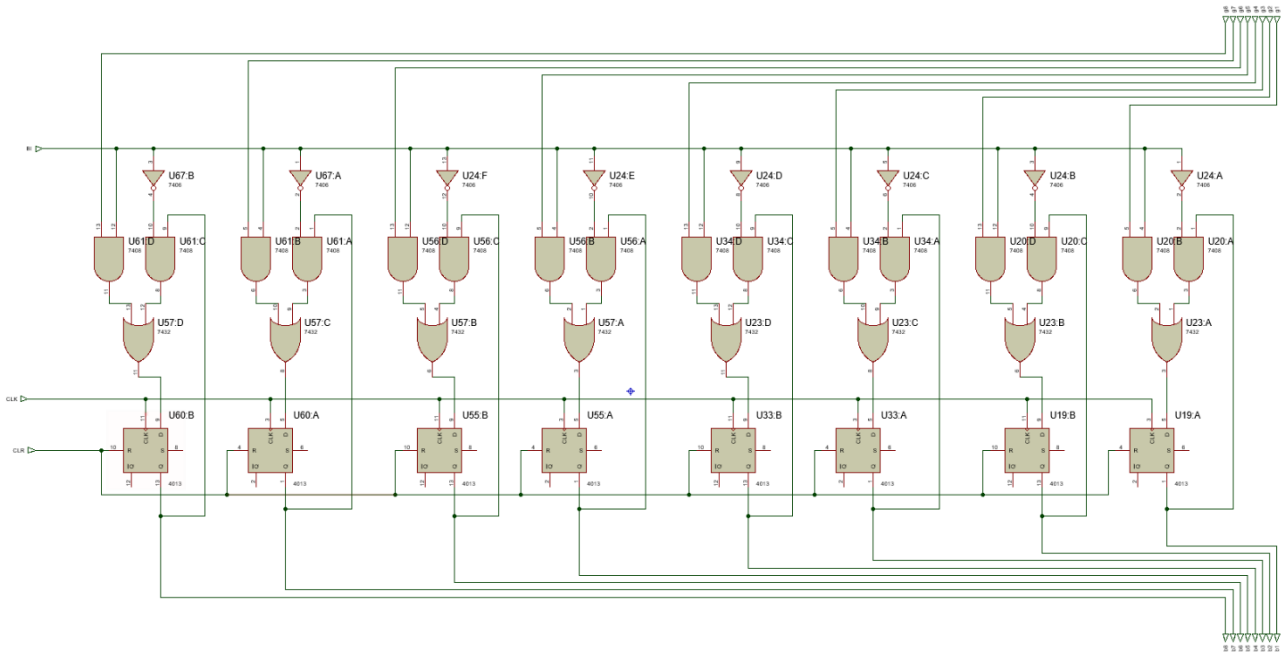
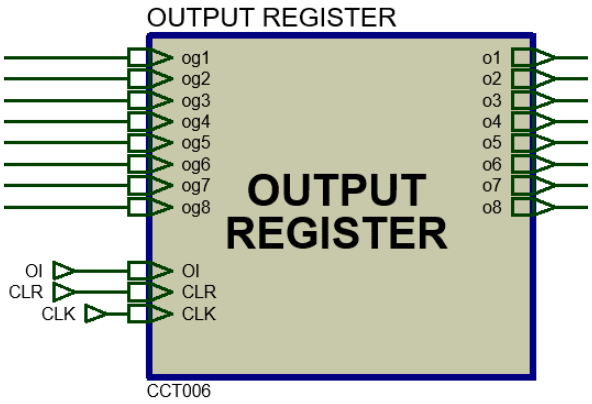


Fig: B- Register Circuit

Output Register

Construction of output register is same as B-register. Here the input instruction is OI which act as the Shift/Load for the register. That is when OI is High then data comes to output register from the accumulator through bus. And when it is Low the register holds the previous values. This is also an 8-bit register. The output register after receiving data from the accumulator sends to the binary to BCD converter module.



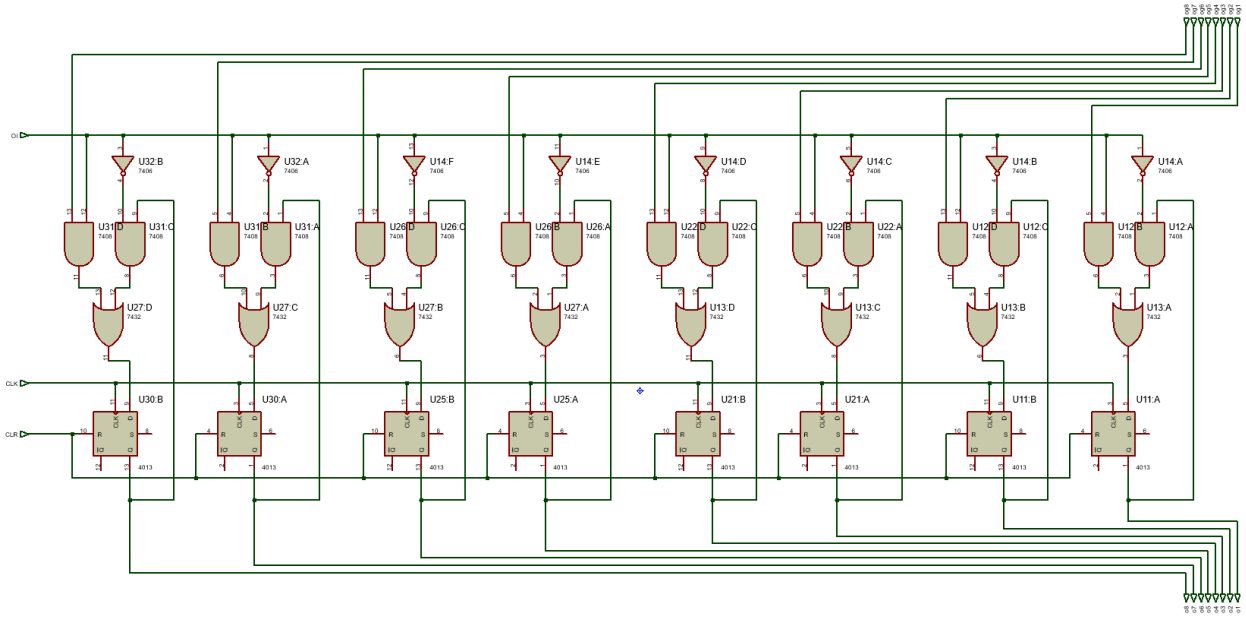


Fig: Output Register Circuit

RAM (16×8BIT)

To design the ram, we follow the basic construction of ram which is a combination of S-R latch and some basic gates like AND, OR, NOT.

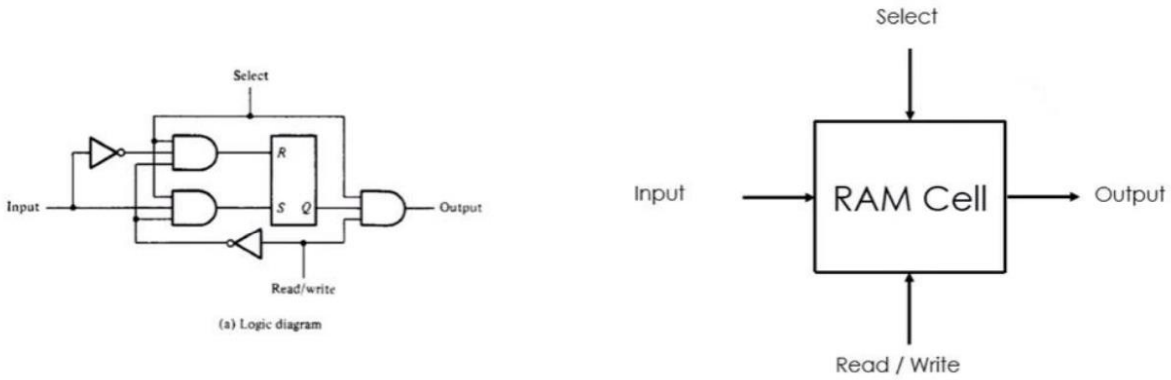


Fig:1-bit ram unit

At first, we try to build 1 BIT ram with S-R by using an IC which has 4 S-R latches (IC-4043). For optimizing the process, we use all inputs of S-R latch IC and make a 4 BIT ram in a subcircuit

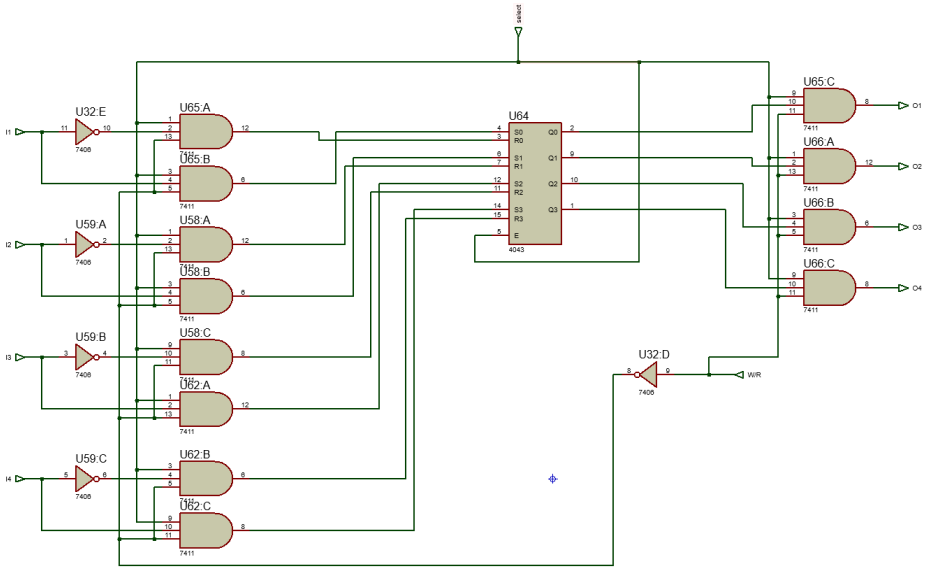


Fig:4-bit ram unit

Then we build a whole 8-bit ram block which has different data lines (8 I/P and O/P), select line (select whole 8-bit block together). Basically this 8-bit ram block is created by merging the 2 4-bit ram with appropriate connection then putting them in a subcircuit.

As we are designing 8-bit SAP, these 8-bit block stores a whole 8-bit data and we can read and write 8-bit data there for further purposes by selecting proper address line.

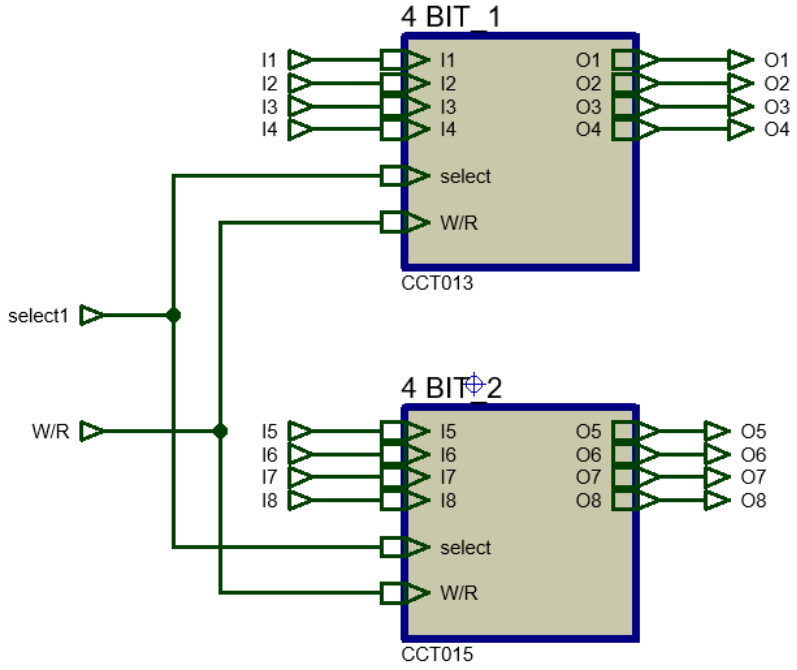


Fig:8-bit ram block with single select line

We follow the basic principle for design ram which is given below:

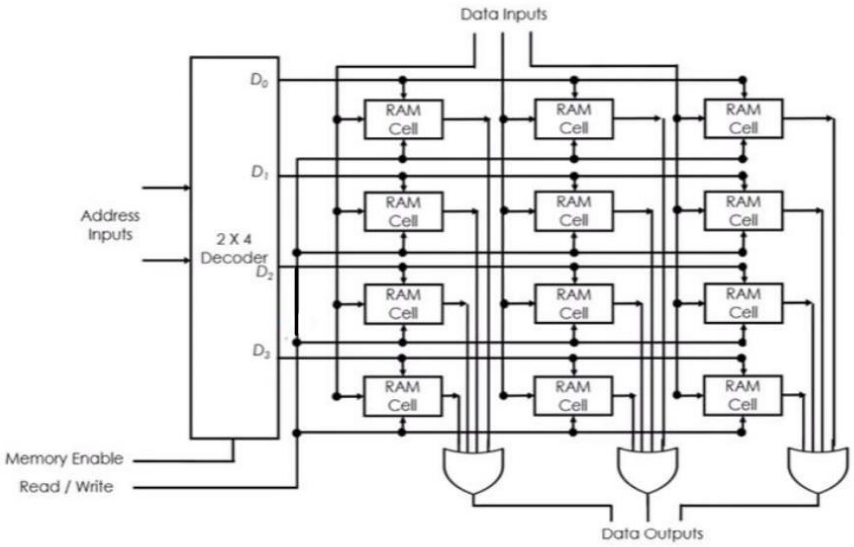


Fig:4×3-bit ram unit

Then, we design a subcircuit for 8×8 bit by using 8 8bit block or 8 8 bit subcircuit. This sub circuit has 8 data lines, 8 selection lines, 1 read and write line. All output lines of same number is connected with or gate.

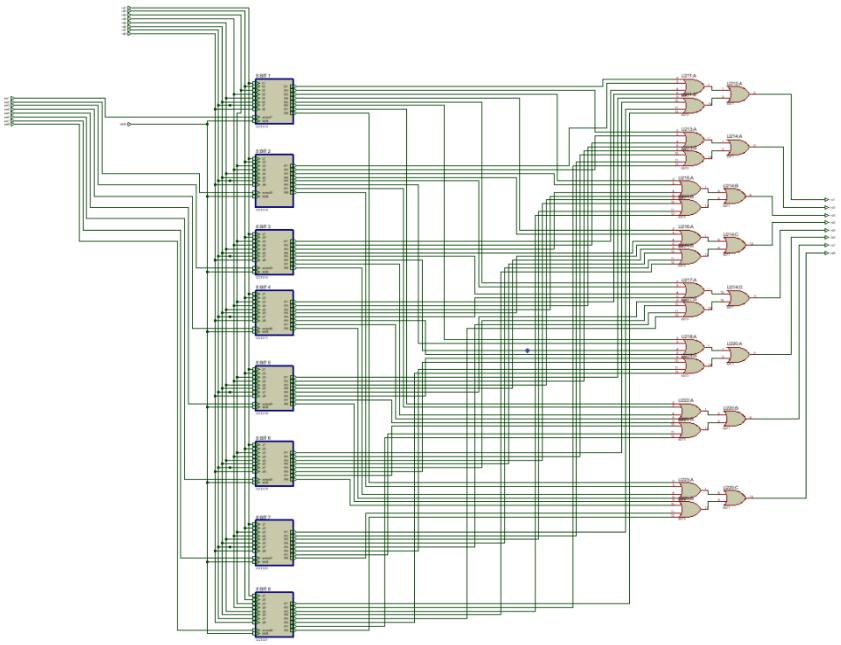


Fig:64-bit (8×8) ram unit

Finally, we use 16 8-bit ram block to design the whole 16×8 bit ram. To select each 8-bit ram block we use a 4 to 16 line decoder (4 to 16 line decoder is built with 2 3 to 8 line decoder by using proper connection).The input of address line is connected to the address selector.

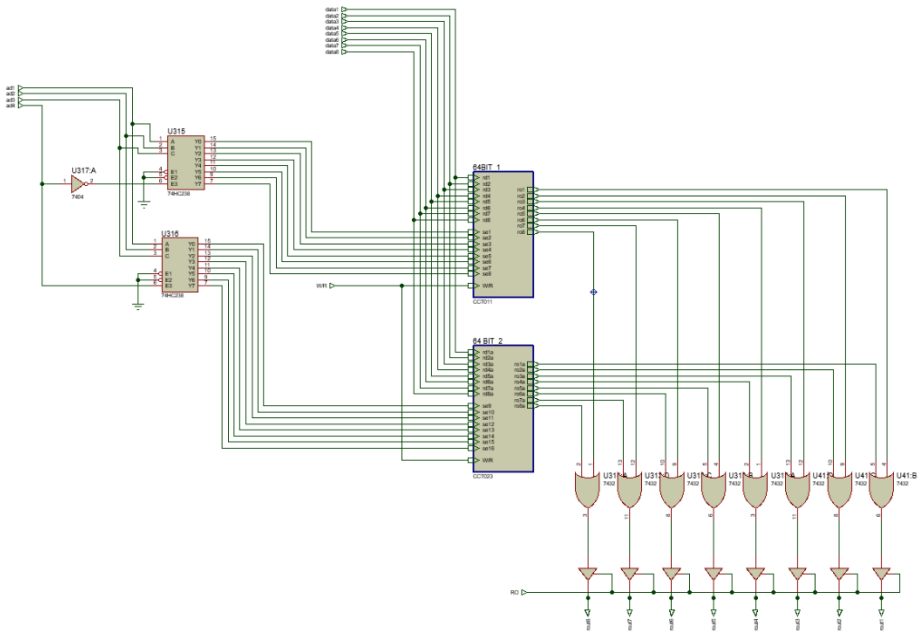


Fig: 16×8-bit ram unit

The tri-state buffer has been used in the output lines of ram for the performing the instruction RO. Thus, we build a required ram for 8-bit SAP design.

Ref.: To design the ram the basic knowledge and all the basic screenshots are taken from EEE-4307 Lecture

Arithmetic and Logic Unit (ALU):

The motto of using this module is to operate binary addition and subtraction. For operating the addition operation, we have used two 4-bit Binary adder. Then we added some modifications with X-NOR gate to operate subtraction operation with the same module actually.

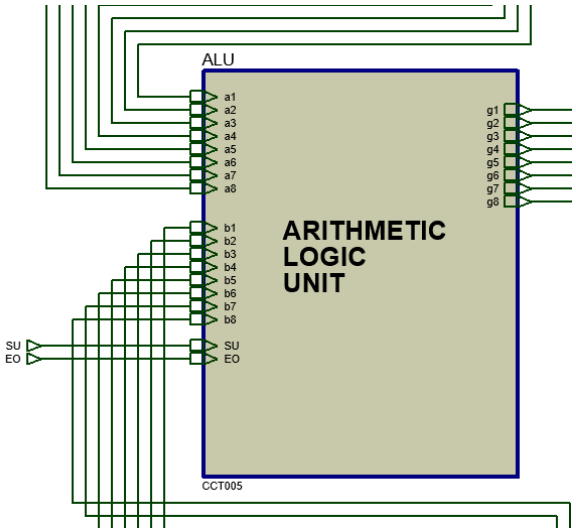


Figure: 01: Module of ALU

Block diagram for Add/ Sub Operation:

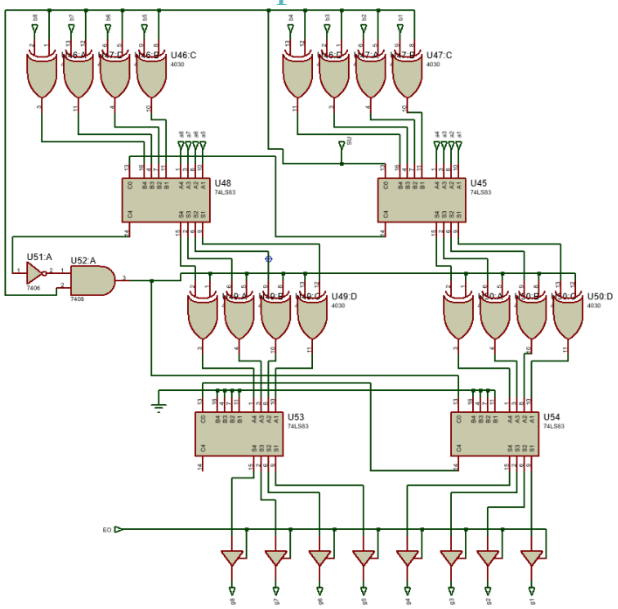


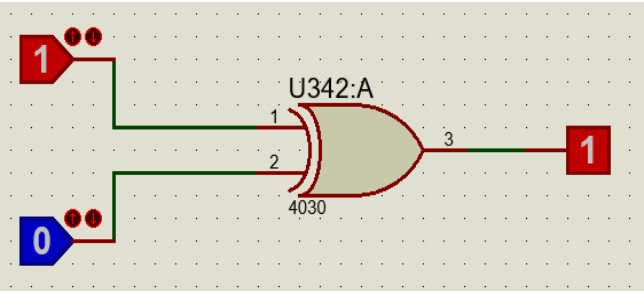
Fig: Adder Subtractor Circuit

Addition Operation:

Suppose, A and B are two-8bit number. When add operation is in-progress SUB/ADD (control bit, SU) is operated in Low condition, so, 1st controllable inverter will be deactivated and the value of A and B will be directly gone to 8-bit Adder. As SUB/ ADD = 0, so, the o/p of the Adder will not be influenced by 2nd 8-bit binary adder and the output will be generated by O1 to O7

Subtraction Operation:

In case of subtraction, SU will be activated, so the operation of controllable inverter will be in-progress. But to discuss the sub operation first we will have to analyze the operation of Basic operation of X-NOR gate.



INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Fig: X-NOR gate with truth table Truth Table

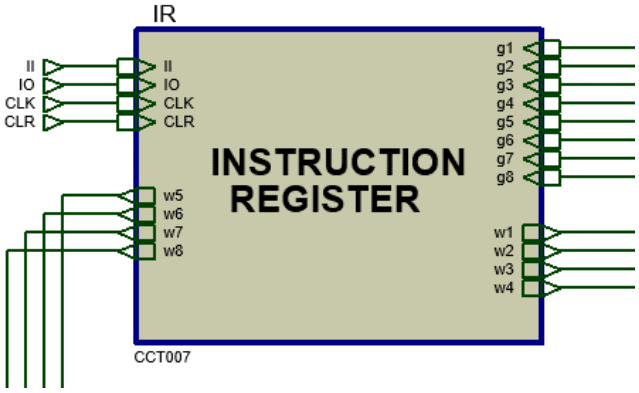
From the truth table we have seen that if one of the inputs is fixed by Low input the output will be similar to the 2nd input. On the other hand, if one of the inputs is fixed by 1(High input) then the output will be inverted with respect to 2nd input. While Sub operation is in progress one of the inputs will be Fixed by giving an input one through SU pin. As one of the inputs of X-OR gate is 1, so whatever the value is given to 2nd input will be inverted and SU is connected with CO of 2nd Adder, So, 1 will be automatically added to the inverted B output. Ultimately, 2's complement is being done.

If the value of A is greater than B then then C_{out} will be 1 and the output of the AND gate will be 1(As SUB=SU= 1 already). So, the output of 1st Adder will be gone through another controllable inverted for the second time and then the value will be shown as output. Actually, whole process is implemented based on the concept of "Subtraction with 2's complement".

Finally, the control bit EO (through a Tri state buffer) will decide when the output of ALU will be put onto the bus.

Instruction Register

Instruction register is also an 8-bit parallel in parallel out shift register. It is almost similar to other resisters used in our SAP 1. The difference is in the output part. Here II is the instruction that allows data to enter from bus. This acts as the Shift/Load of the selection circuit (explained in MAR portion). But here first nibble of the output goes directly into controller and the second nibble through tristate buffer which gets activated by IO goes to the bus. And then from bus to MAR again.



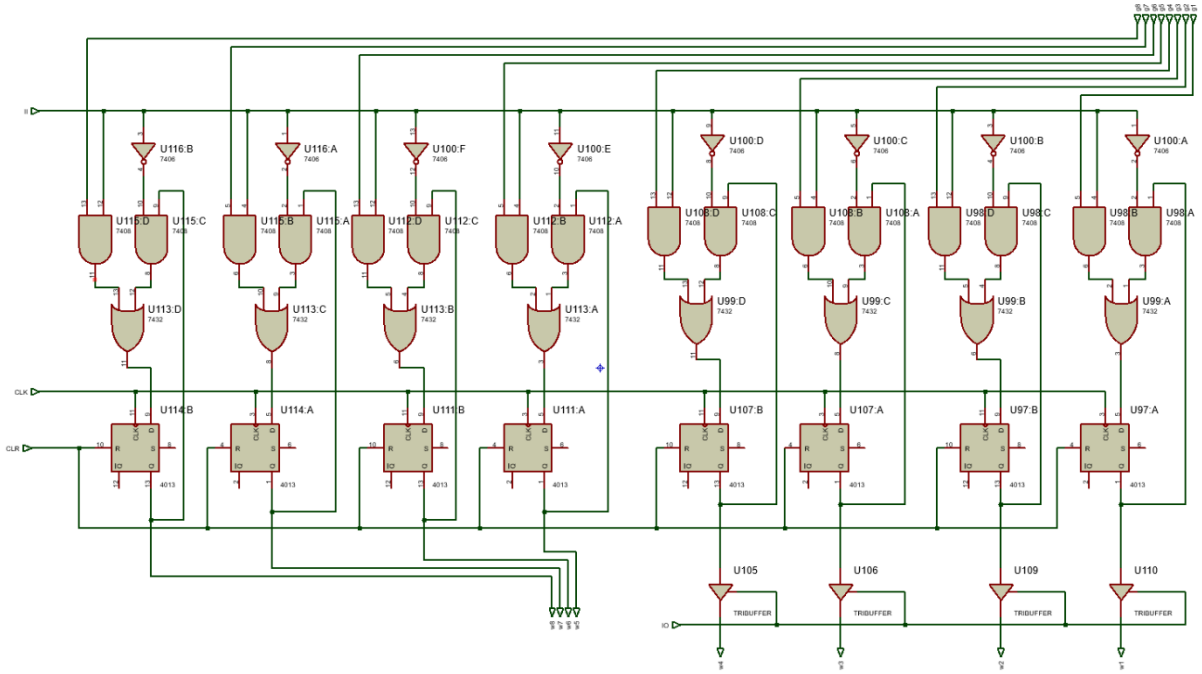


Fig: Instruction Register

Controller/Sequencer

Controller is like the brain of SAP-1 from where all the necessary micro instructions are sent to different modules according to T-states. We can divide it to three parts. These are as follows:

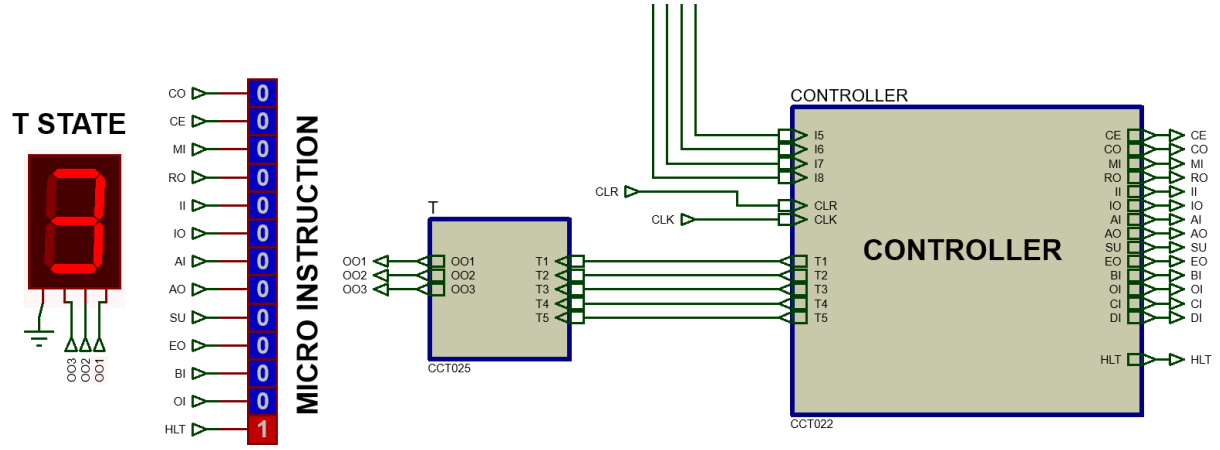


Fig: Controller Subcircuit with T-state and Micro Instruction

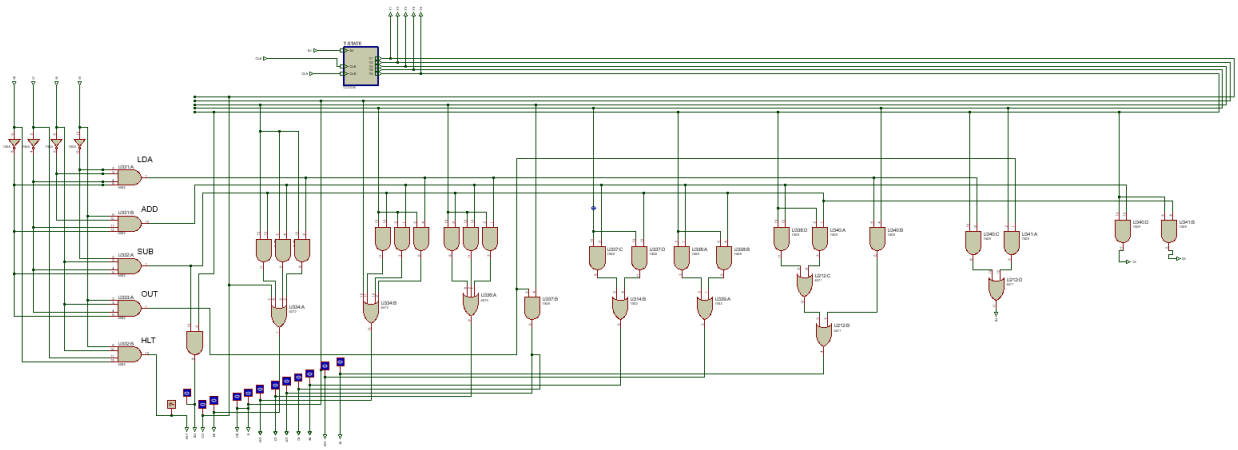
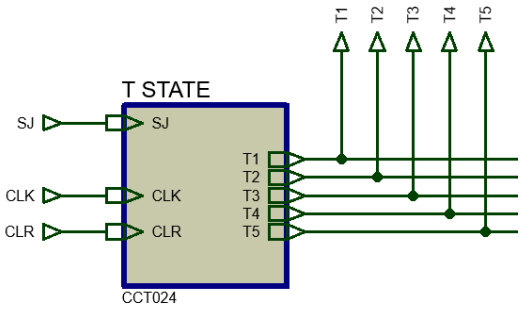


Fig: Controller/ Sequencer Circuit

T-State counter

This consists of two parts: A four-bit up counter and a modified decoder circuit. The four-bit counter counts gradually and sends data to the decoder circuit which gives T-States as outputs and as soon as it reaches the binary equivalent of five it gets resets. There is one extra instruction SJ to reset the counter which gets activated when there is need to skip T states. These can be realized better from the following table:



Counter	Decoder
0000	T1
0001	T2
0010	T3
0011	T4
0100	T5
0101	Reset

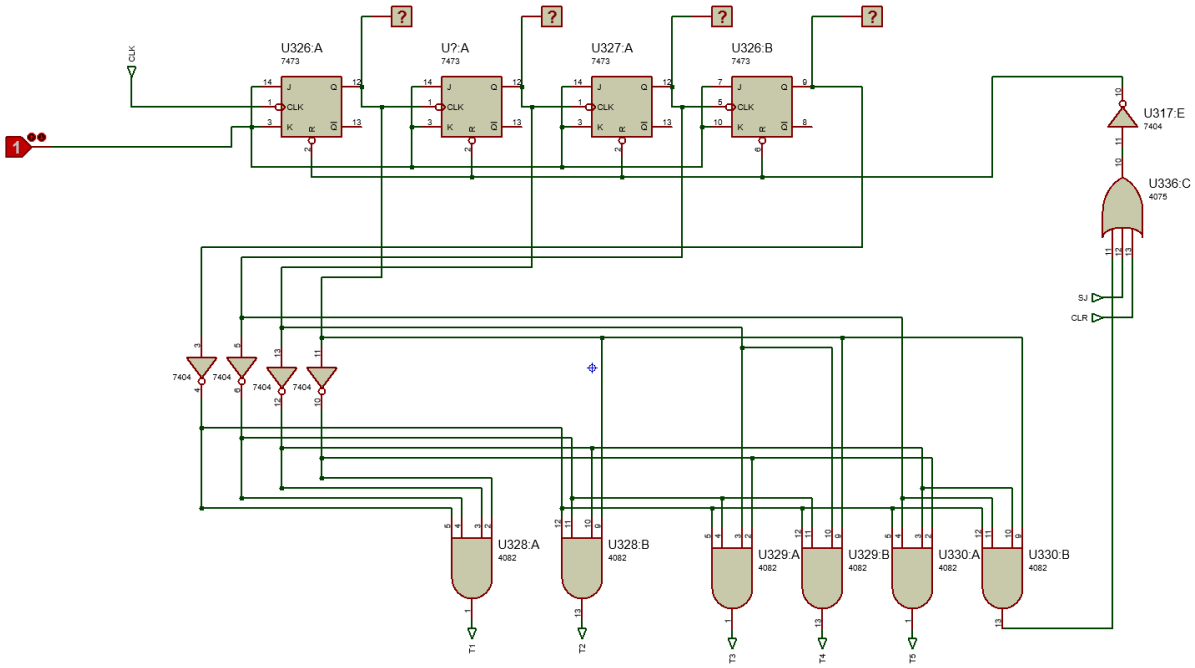


Fig: T- State Counter

Macro Instruction

To generate the Macro Instruction, we made a modified decoder circuit similar to the one used in the T state counter.

I8	I7	I6	I5	Decoder
0	0	0	0	LDA
0	0	0	1	ADD
0	0	1	0	SUB
0	0	1	1	OUT
0	1	0	0	HLT

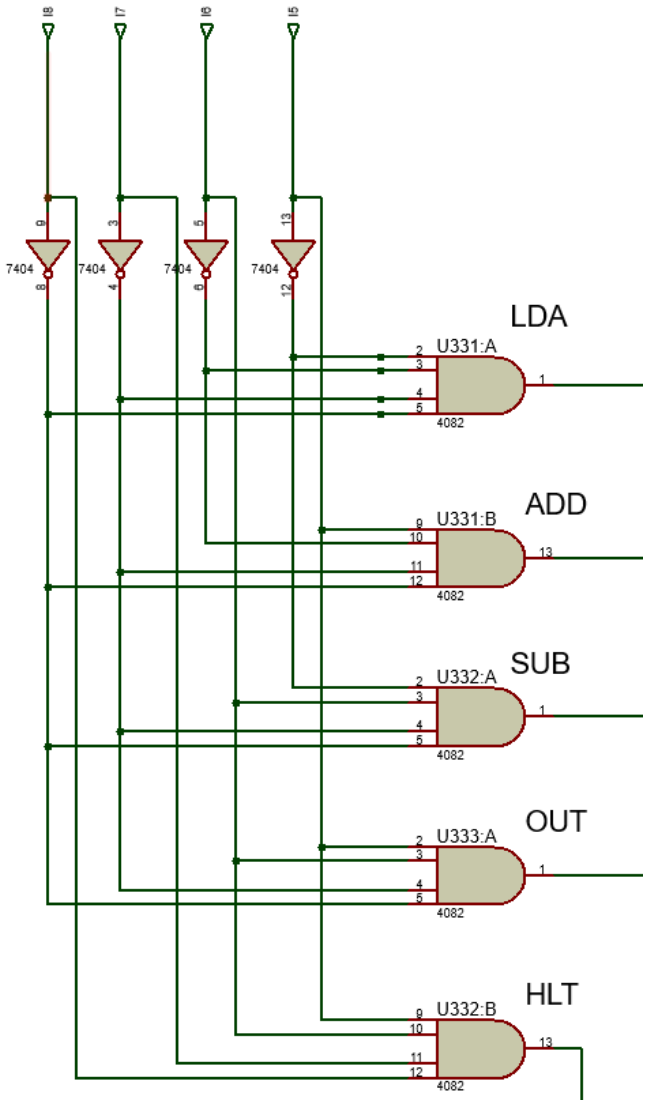


Fig: Macro Instruction Circuit Diagram

Micro Instruction

There are in total of 16 instructions we have incorporated in out SAP-1 project. The extra instructions are SJ SI and DI. The instructions are activated according to the following table.

	LDA	ADD	SUB	OUT	HLT
T1	CO MI	CO MI	CO MI	CO MI	CO MI
T2	CE RO II	CE RO II	CE RO II	CE RO II	CE RO II
T3	IO MI	IO MI	IO MI	AO OI	STOP CLK
T4	AI RO	RO BI	RO BI	SJ	
T5	SJ	EO AI CI	EO AI SU DI		

These instructions are discussed below:

CO: When this is active, value from program counter goes to bus.

Fig: CO instruction Logic Diagram

CE: When this is active, program counter gets activated and value is incremented inside it.

Fig: CE instruction Logic Diagram

MI: This instruction allows MAR to collect data from the bus.

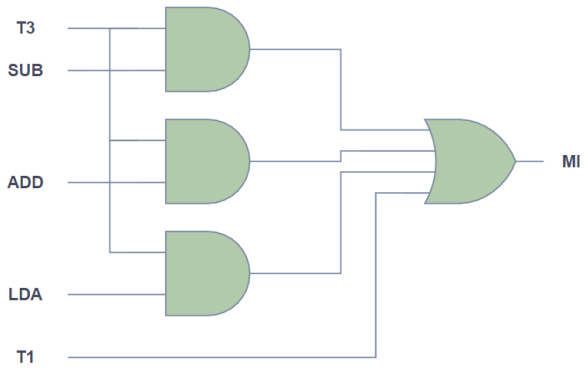


Fig: MI instruction Logic Diagram

RO: This instruction allows RAM to send 8-bit data to the bus as per requested address.

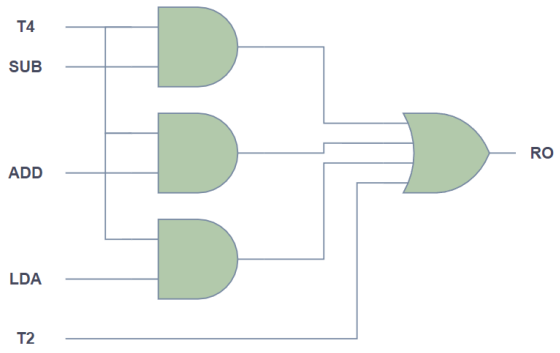


Fig: RO instruction Logic Diagram

II: This instruction allows instruction register to get data from the bus.

Fig: II instruction Logic Diagram

IO: This allows instruction register to send LSB 4-bit of the 8-bit it receives to the bus.

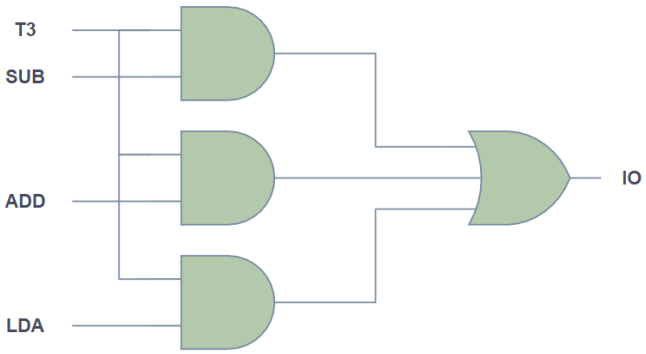


Fig: IO instruction Logic Diagram

AI: When AI is HIGH, accumulator gets data from the bus.

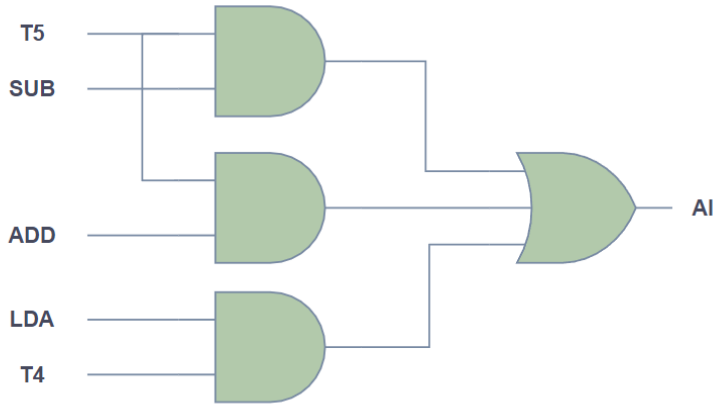


Fig: AI instruction Logic Diagram

AO: This instruction allows data to flow from accumulator to got to the bus.

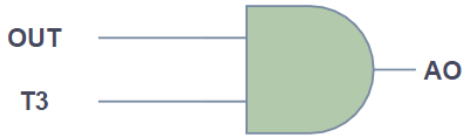


Fig: AO instruction Logic Diagram

CI: When CI is ON, the added value of the first two number is sent to the C register.

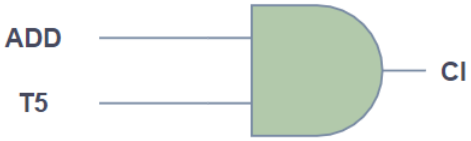


Fig: CI instruction Logic Diagram

SU: This instruction allows ALU to perform subtraction operation.

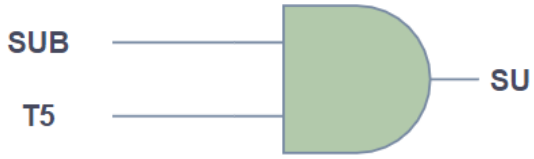


Fig: SU instruction Logic Diagram

DI: The value to be subtracted and to be compared with the added value is sent from B register to D register.

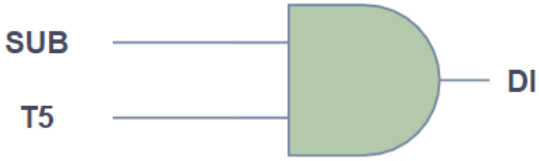


Fig: DI instruction Logic Diagram

EO: This enables ALU to send its data to the bus.

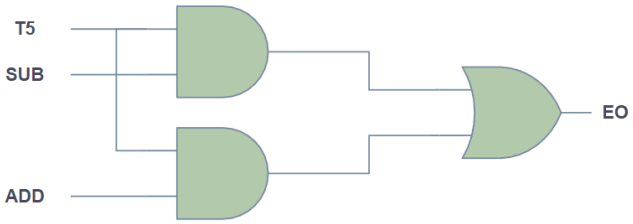


Fig: EO instruction Logic Diagram

BI: This enables B-register to get values from the bus.

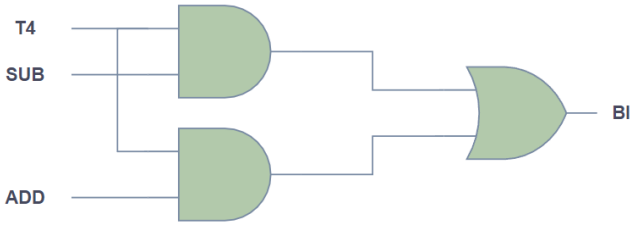


Fig: BI instruction Logic Diagram

OI: This instruction allows data to flow from output register to the display unit.

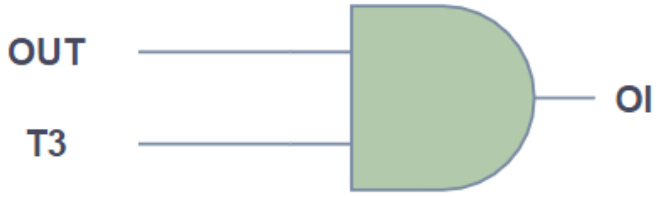


Fig: OI instruction Logic Diagram

SJ: This instruction is to jump T states. We can see from the above table that Load state doesn't need 5th T state and Out state doesn't need 4 and 5th T state. So, to skip these states we introduced SJ.

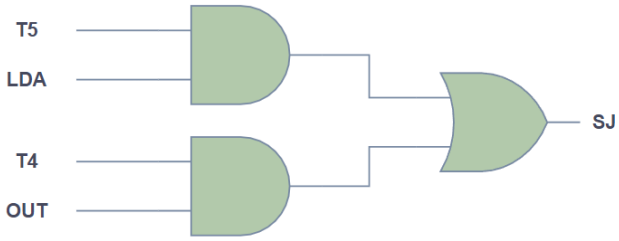


Fig: SJ instruction Logic Diagram

HLT: When HLT is active all the activities of the SAP-1 are stopped by stopping the clock.

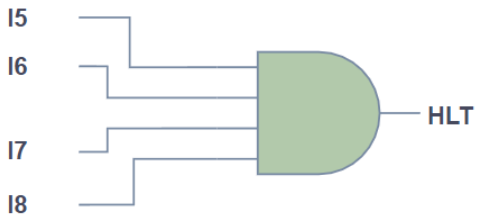
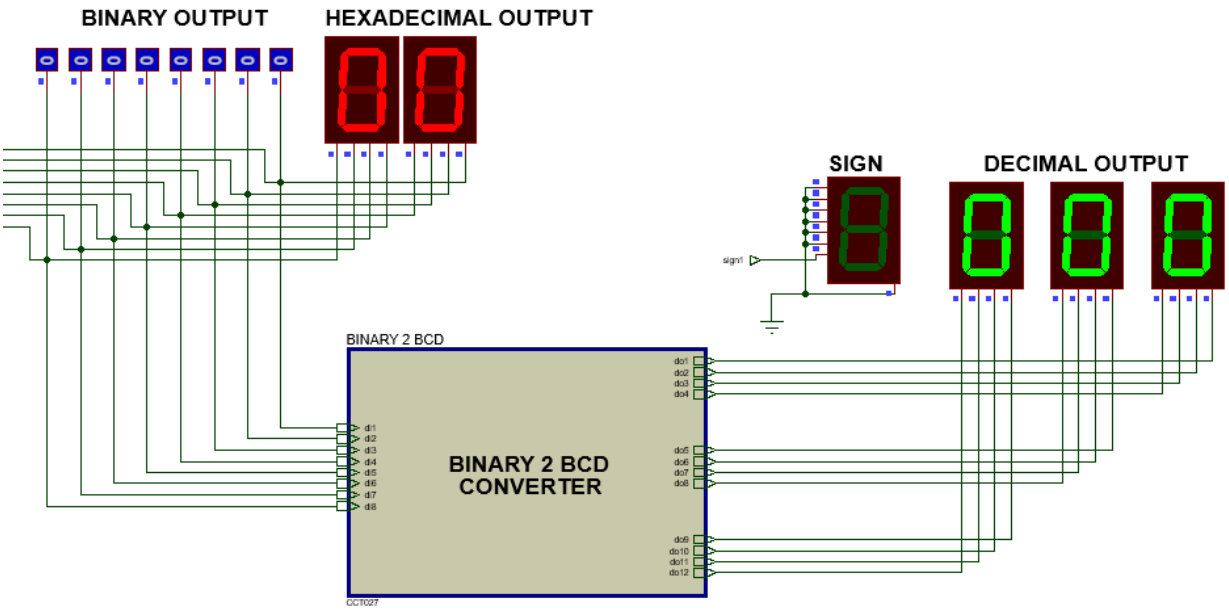


Fig: HLT instruction Logic Diagram

Binary Display



Here from output register we get 8-bit binary data. These are sent to binary to BCD converter module. The Binary to BCD converter is constructed in the following way.

Block Diagram

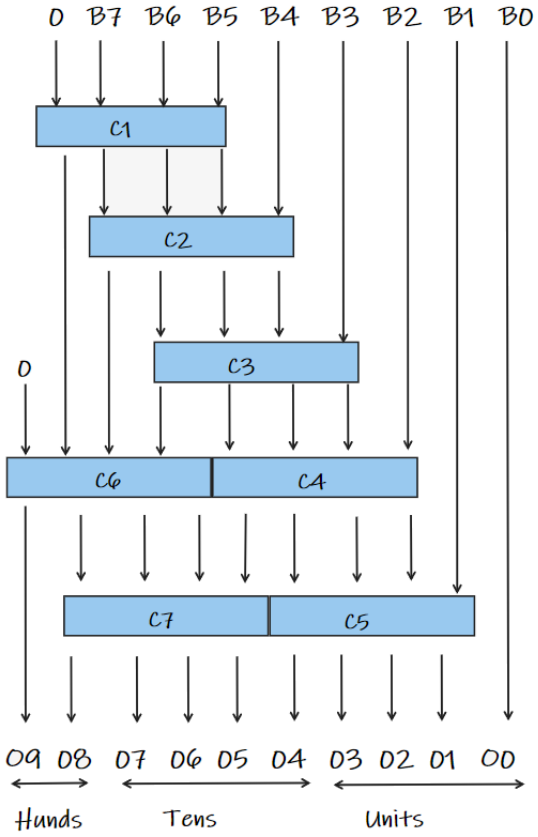
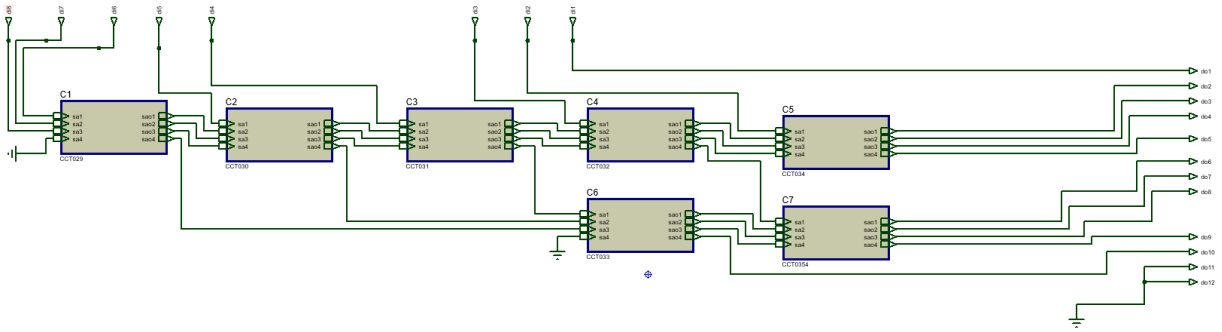


Fig: Binary to BCD converter logic diagram

Here first we create units, tens and hundreds column of 4 bit size and shift our data to the left. When the data reach value equal to or greater than 5 we add 3 to it. We keep on continuing this until all the data are shifted.



Each of these subcircuit are combinational circuit to excute the add 3 operation which is, when the four bit data is greater than four it adds 3 otherwise keeps the data as it is.

Truth Table for the C blocks

A3	A2	A1	A0	S3	S2	S1	S0
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

Boolean Expression found using K-Map

S3=A2A0 + A2A1 + A3

S2= A2A1'A0' + A3A0

S1=A2'A1 + A3A0' +A1A0

S0=A3' A2' A0 + A2 A1 A0' + A3 A0'

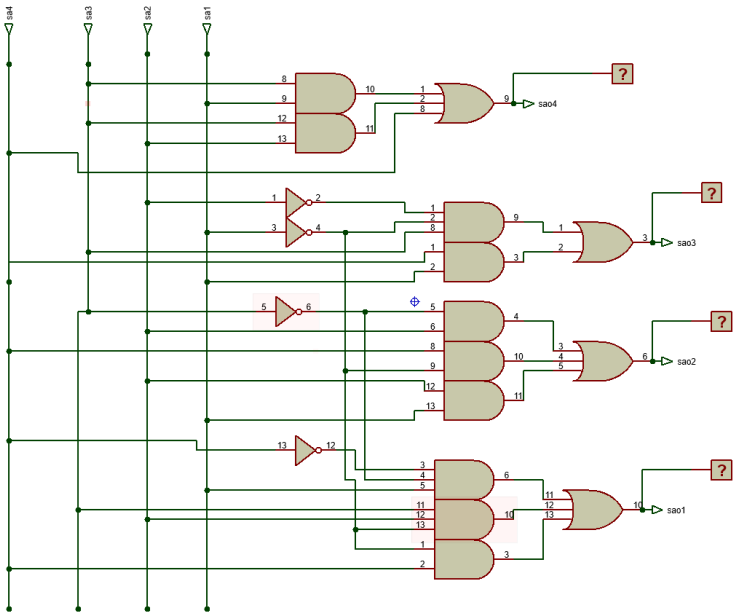
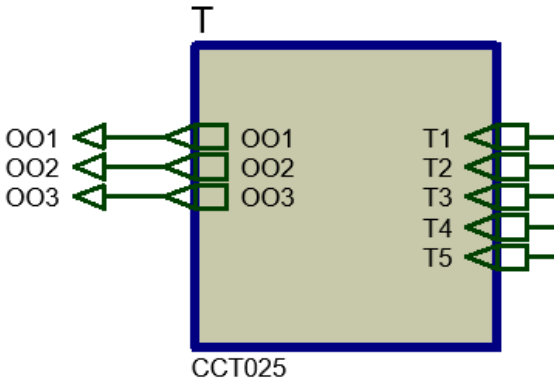


Fig: C block circuit diagram

T- State Display

This subcircuit is to display at which T state we are currently in. It's a simple combinational circuit which we got from solving the following truth table using K-maps and deriving the Boolean expression.



Truth Table

T5	T4	T3	T2	T1	O3	O2	O1
0	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	0	0	0	1	0	0
1	0	0	0	0	1	0	1

Boolean Expression

O3= T5+T4

O2 = T3+t2

O1 = T5+T3+T1

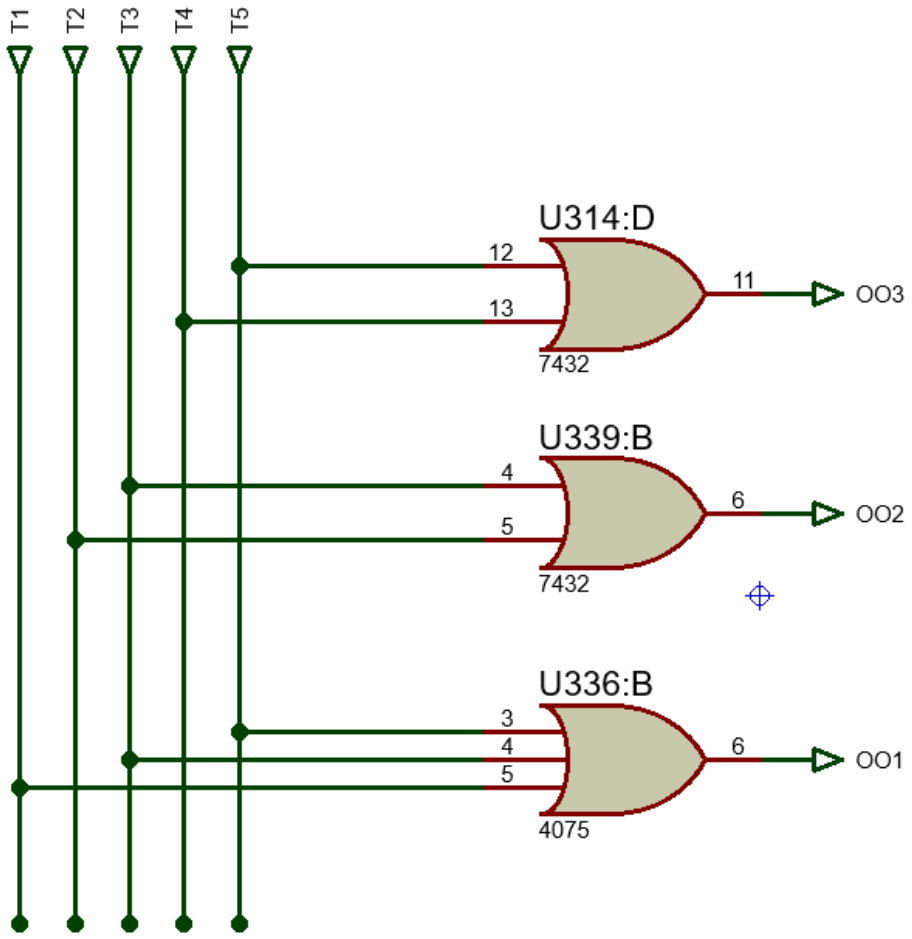


Fig: T- State Display Internal Logic Circuit

Additional Modules

To show the sign of the output number we constructed extra two models. These two are C register and D register. To operate these two registers, we added two extra instructions namely CI and DI. When CI is HIGH C register gets 8-bit data from the ALU. This is the added value of the first two data we gave as input. Then later when DI is HIGH the D register will get 8-bit data from the B register. Then the comparators will compare these two values and if the value is to be negative then it'll send a signal to the extra display we have used to show the negative sign.

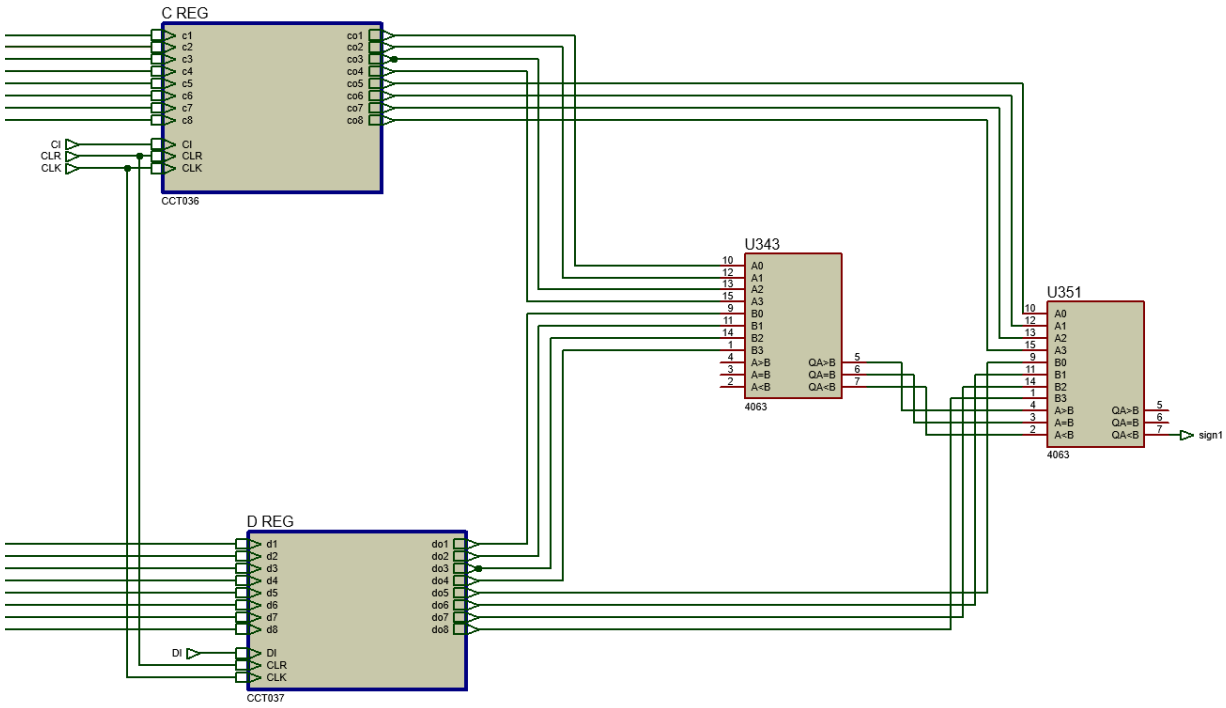


Fig: Module to generate sign bit

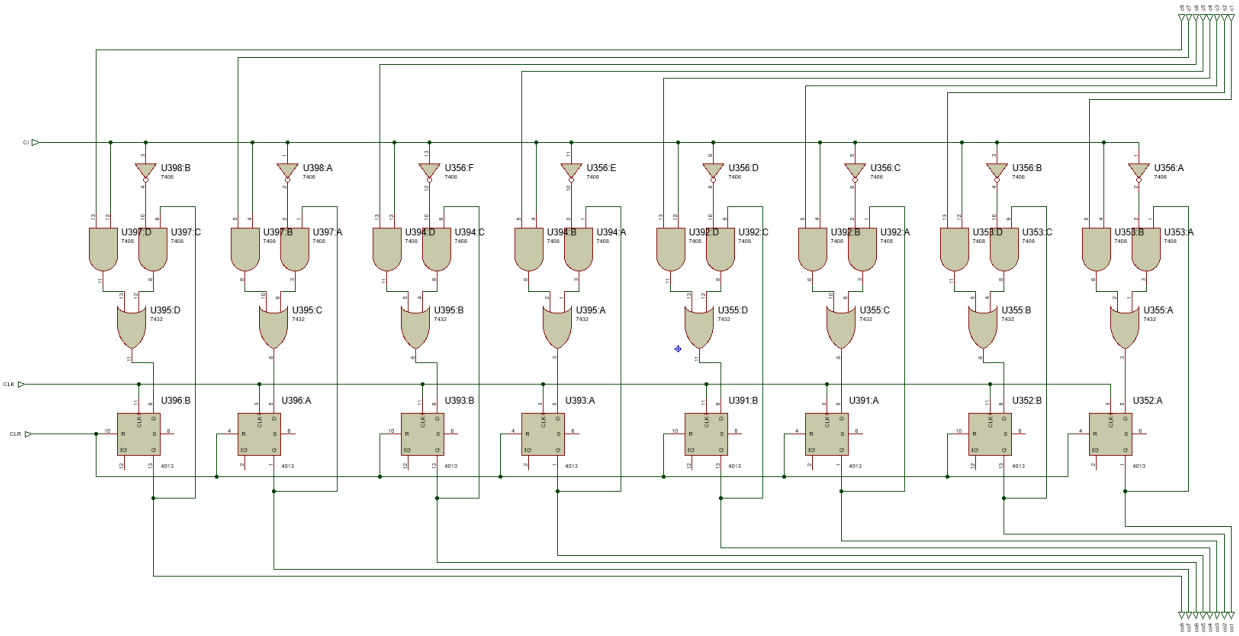


Fig: C register

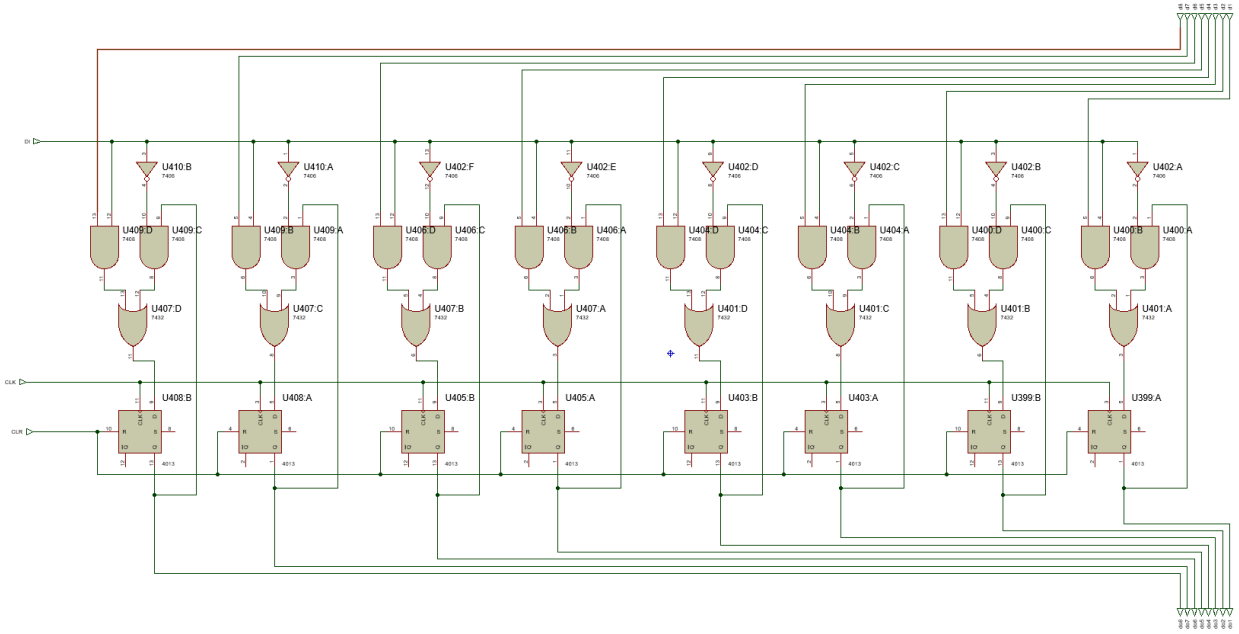
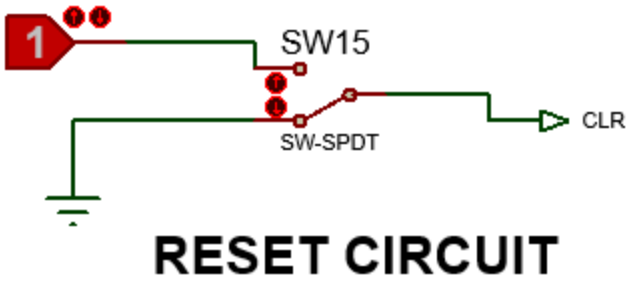


Fig: D register

Reset Circuit



RESET CIRCUIT

This is a simple reset circuit using SPDT switch where one input is connected to High logic state another to ground. So, whenever we need to clear every module in our SAP-1 we connect it to High logic state. That is we have active high clear output.

Complete Project

Block Diagram

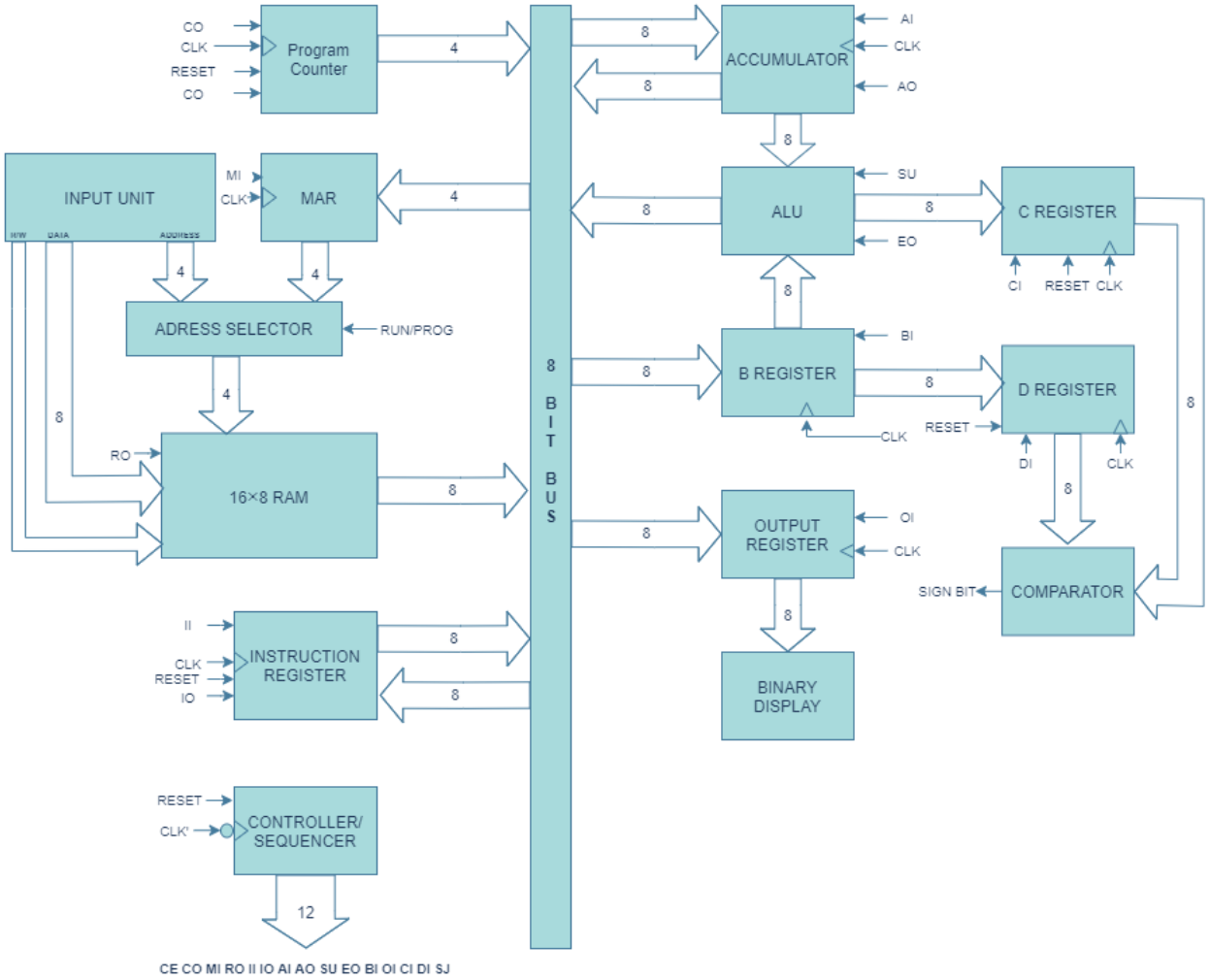


Fig: Block Diagram of the Project

Circuit Diagram

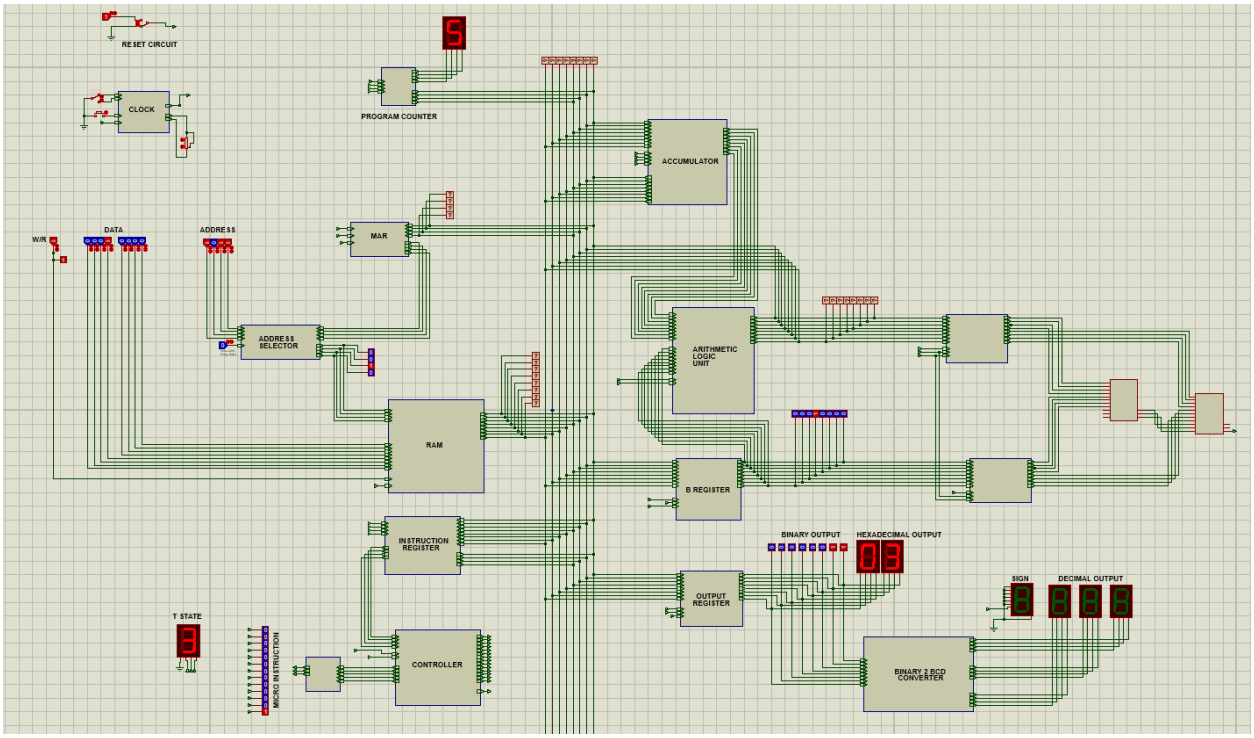


Fig: Proteus Implementation of SAP-1 Computer

Discussion

Outcomes

We built the Simple As Possible (SAP)-1 computer which is a very basic model of a microprocessor. So, doing this project we came to realize the basic mechanism of a microprocessor, how it works, etc. We also understood how various electrical modules interact with each other, how data are processed and stored in electronic devices, how to overcome the problems within their interaction and so on. In a word we took a step forward towards the building of modern computer. We also have implemented quite everything that we have learnt through out the theory courses.

Limitations

Since this project was completed of Proteus Software, we had to face various constraints related to the software. One such barrier was using a definite NOT gate IC 7406. This IC worked for certain modules but failed to function properly for other modules. The most prominent limitation was not to implement the SAP-1 in real life, that is hardware implementation. Through this hardware implementation we could've become more efficient practically and learnt more.

Future Development

There are couple scope to develop what we have made. These can be-

- To reduce T-State from 5 to 4
- Optimize more in terms of gates used
- Use EEPROM instead of using huge combinational circuits.
- Hardware implementation

THE END