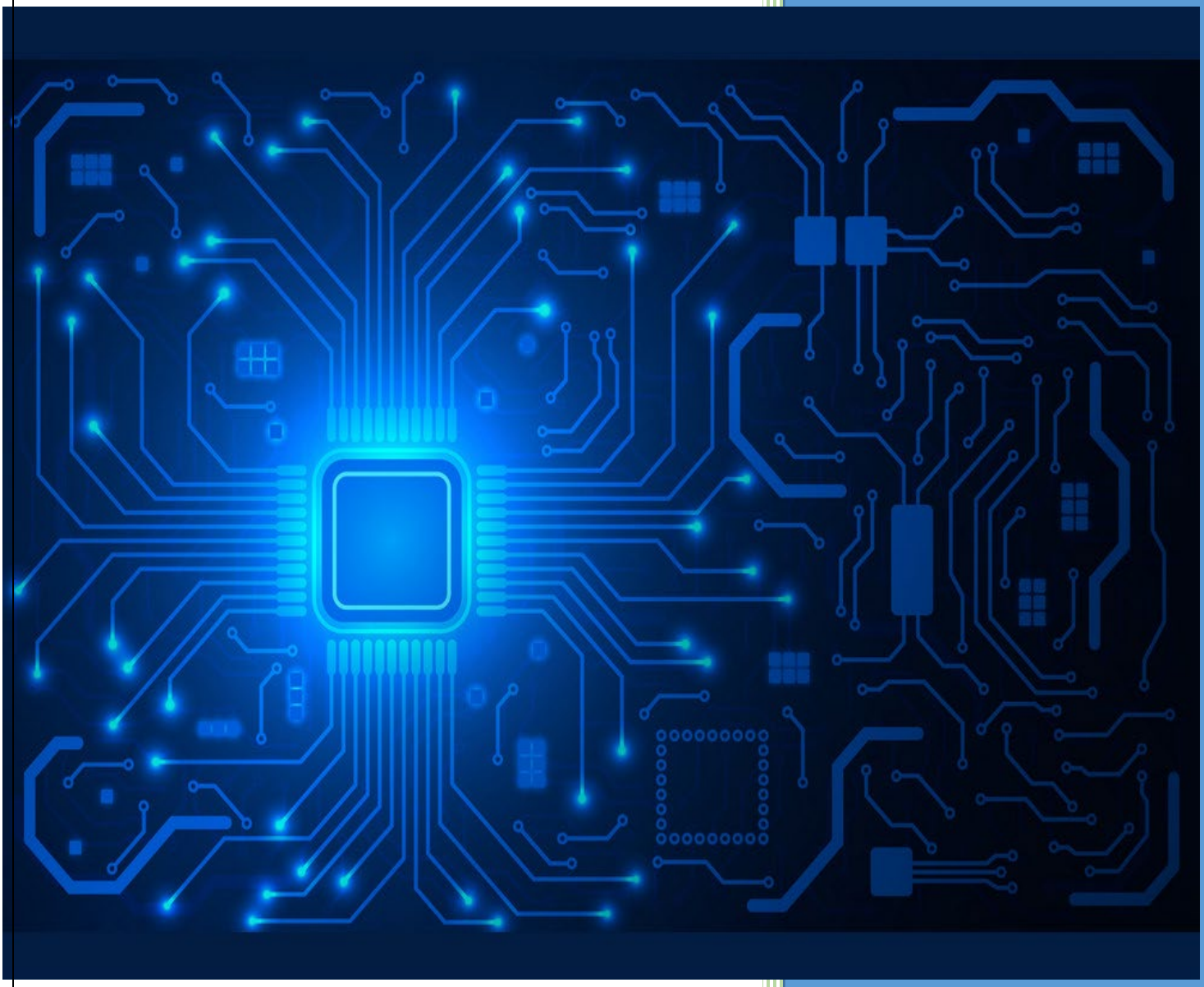
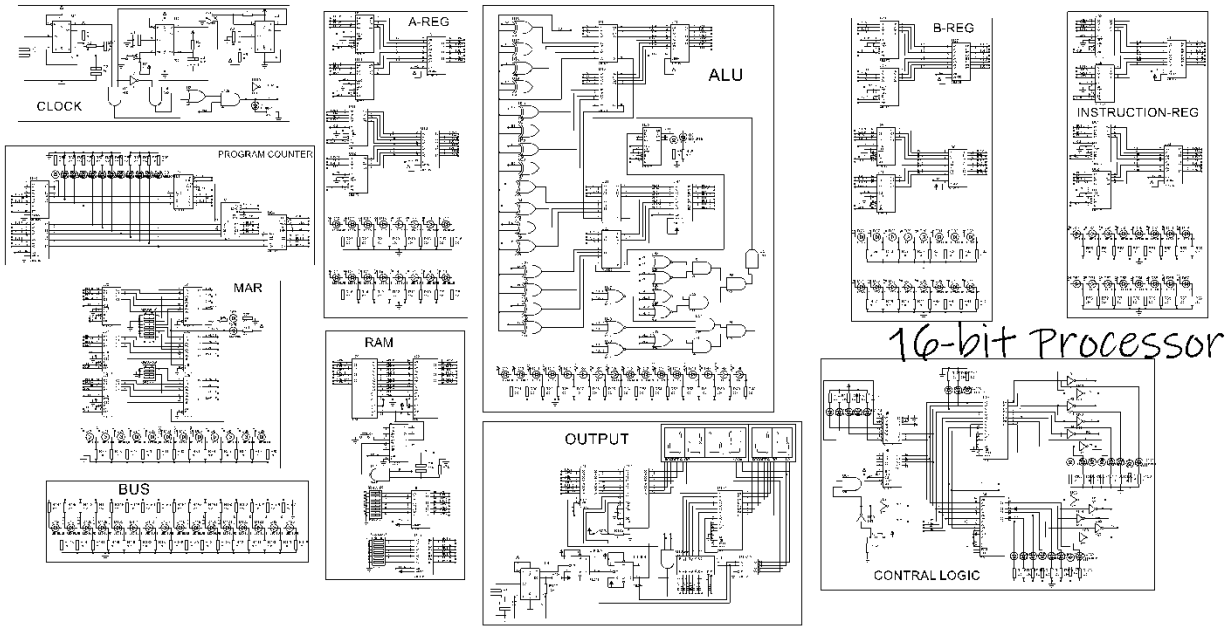
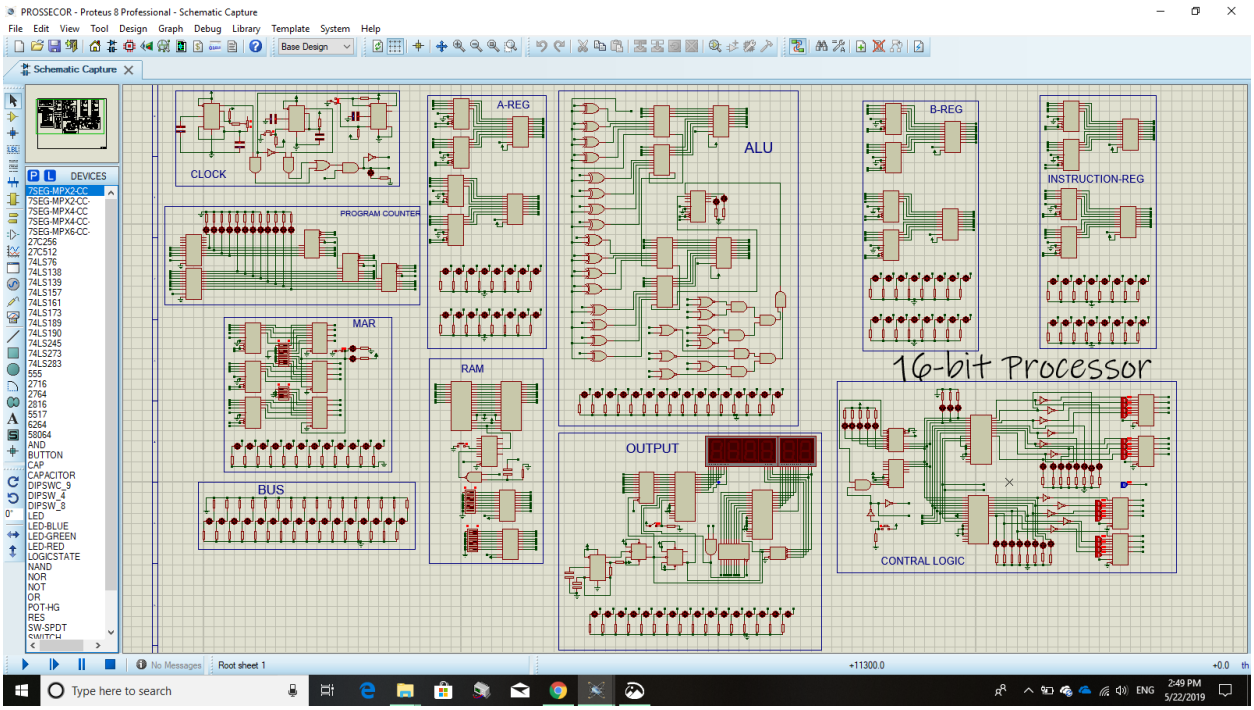
16-bit microprocessor



Overview :

I built a programmable 16-bit computer using only simple logic gates. The 16-bit computer is just able to add and subtract.





Clock module :

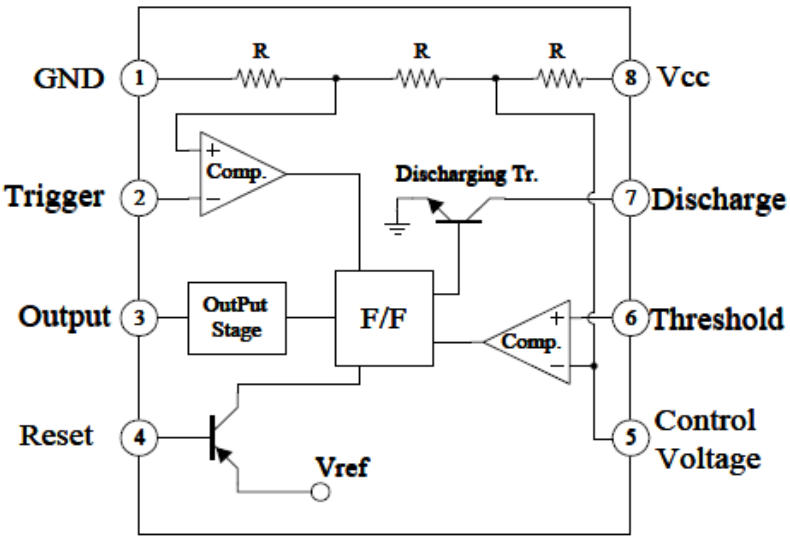
The computer's clock is used to synchronize all operations. The clock we're building is based on the popular 555 timer IC.

Our clock is adjustable-speed (from less than 1Hz to a few hundred Hz). The clock can also be put into a manual mode where you push a button to advance each clock cycle. This will be a really useful feature for debugging the computer later on.

The timer 555ic has 3 mode to generate pulses :

1. Monostable .
2. Astable.
3. Bi –stable.

Internal Block Diagram



1. Monostable Operation

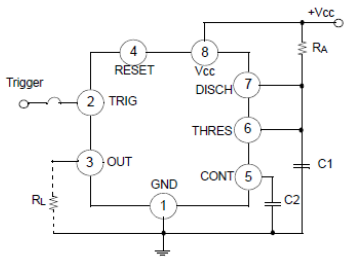
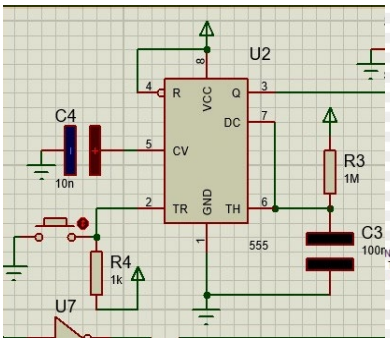
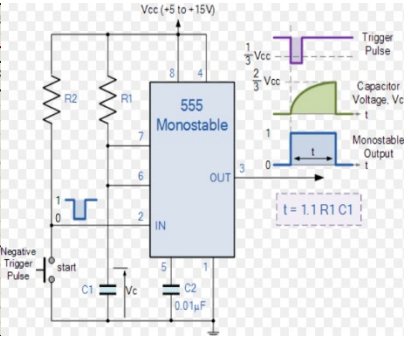


Figure 1. Monoatable Circuit





2. Astable Operation

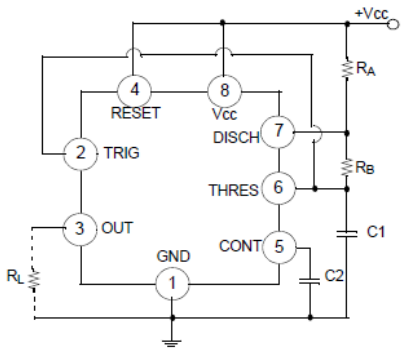
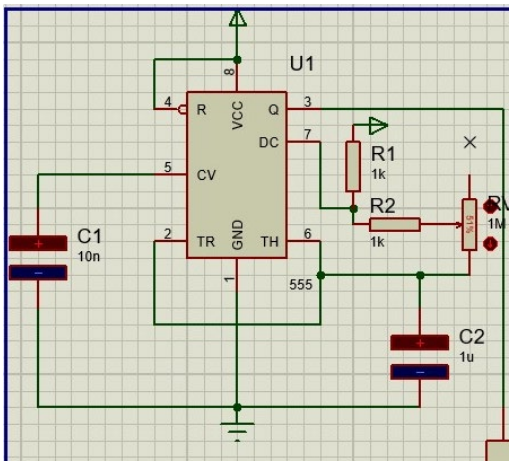
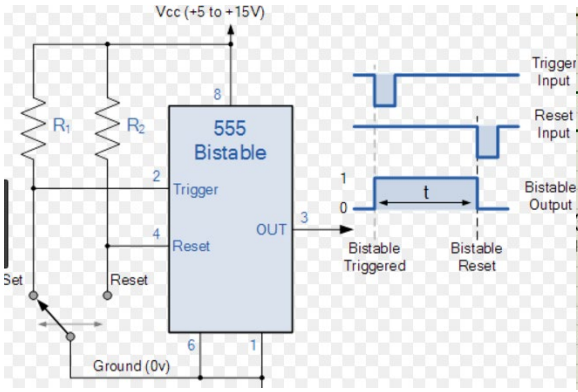
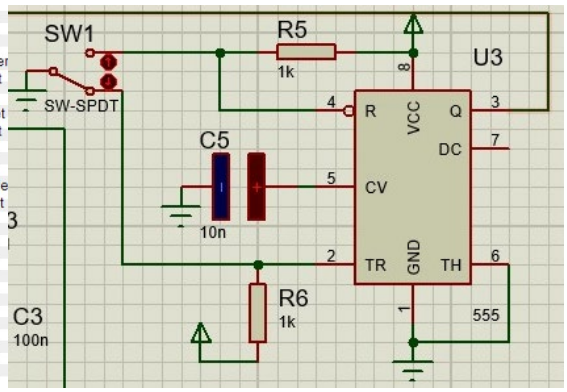


Figure 5. Astable Circuit

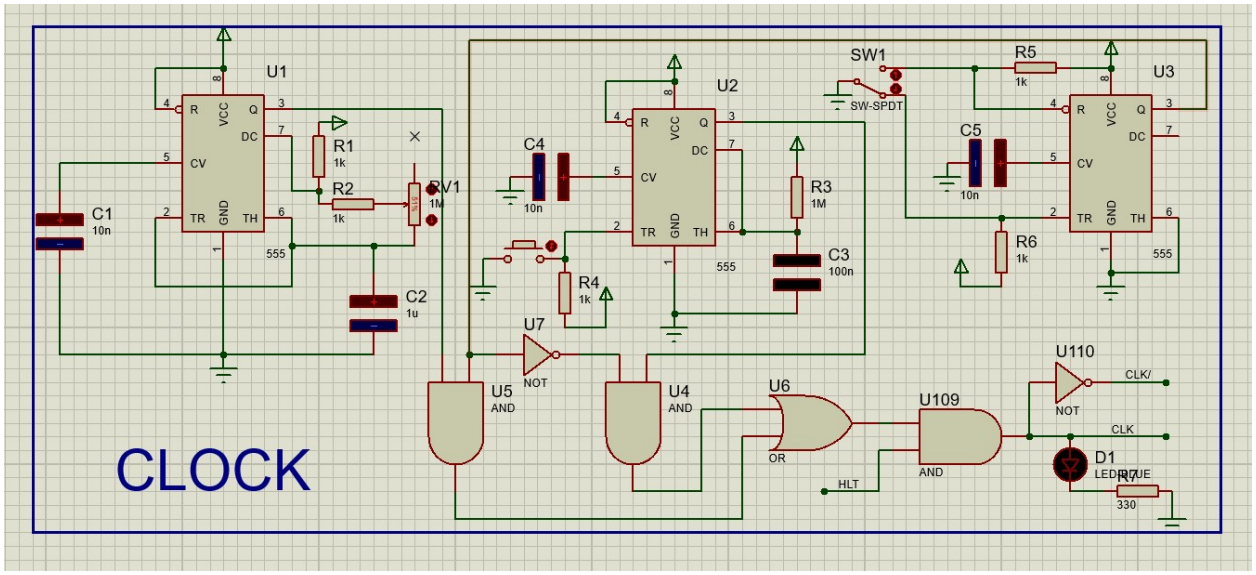


3. Bi –stable.





Using the 3 module we could make our clock for microprocessor like this is schematic :



Note : the label (HLT) coming from control logic (CPU).

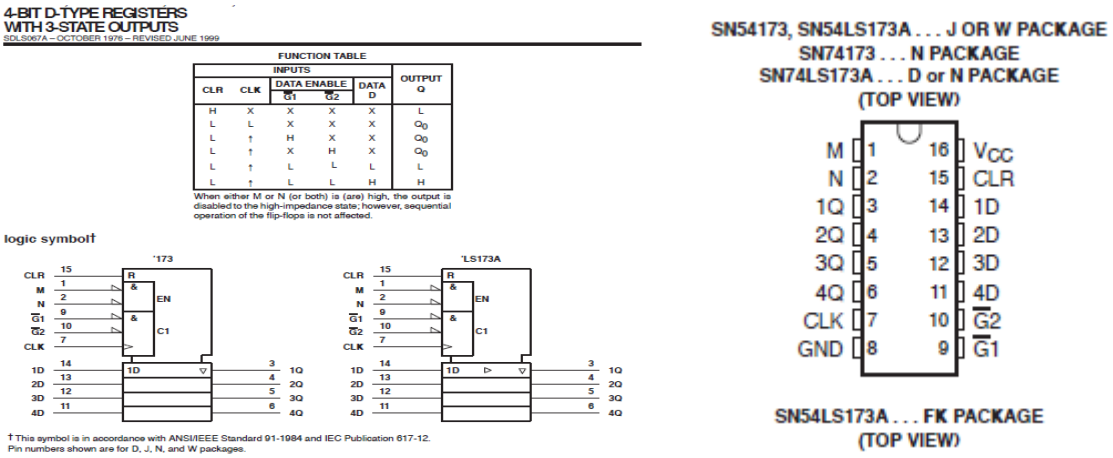
Note : we using logic gate to control in clock manual or by CPU or stop it or get one plus .

Registers :

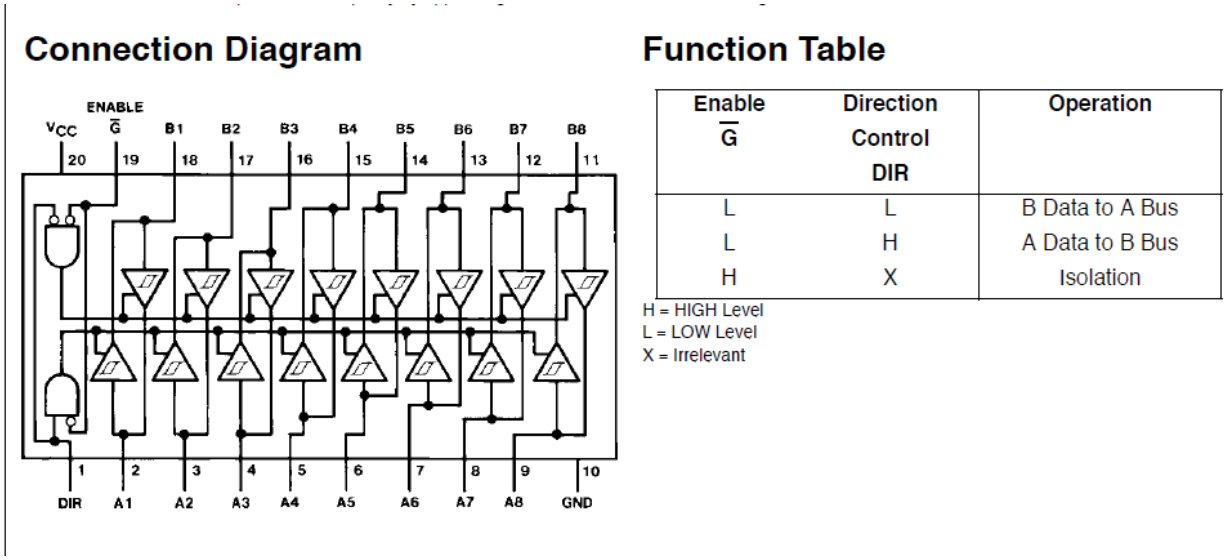
Most CPUs have a number of registers which store small amounts of data that the CPU is processing. In our simple, we'll build three 16-bit registers: A, B, and IR. The A and B registers are general-purpose registers. IR (the instruction register) works similarly, but we'll only use it for storing the current instruction that's being executed.

We used a 4-bits D-type Register to store data and like a function table in datasheet of 74173-IC we control a data enable to store data in register and using Octal Bus Transceiver 74245-IC could control when appear the storing data on the bus.

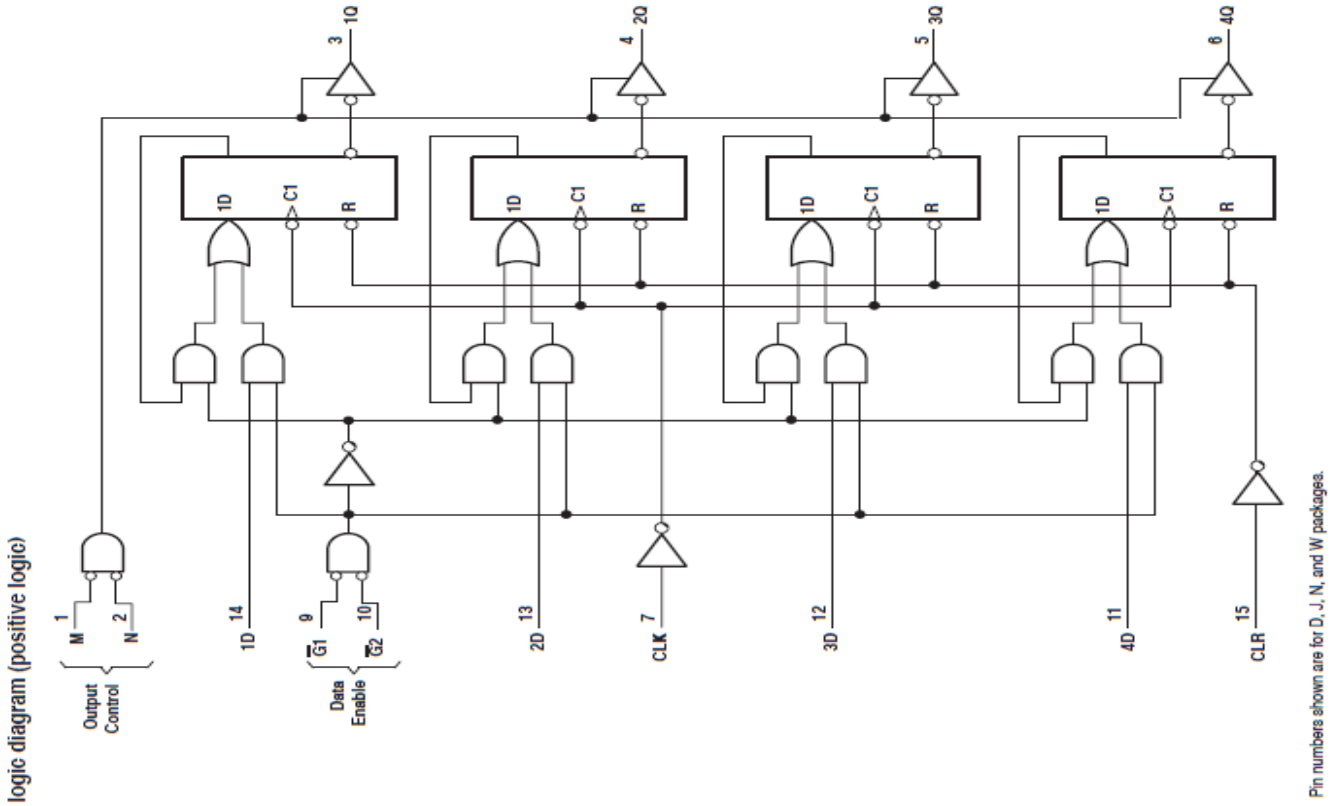
Note : write data on register or reading from it is a (Active Low)pins , put reset pin is (Active High).

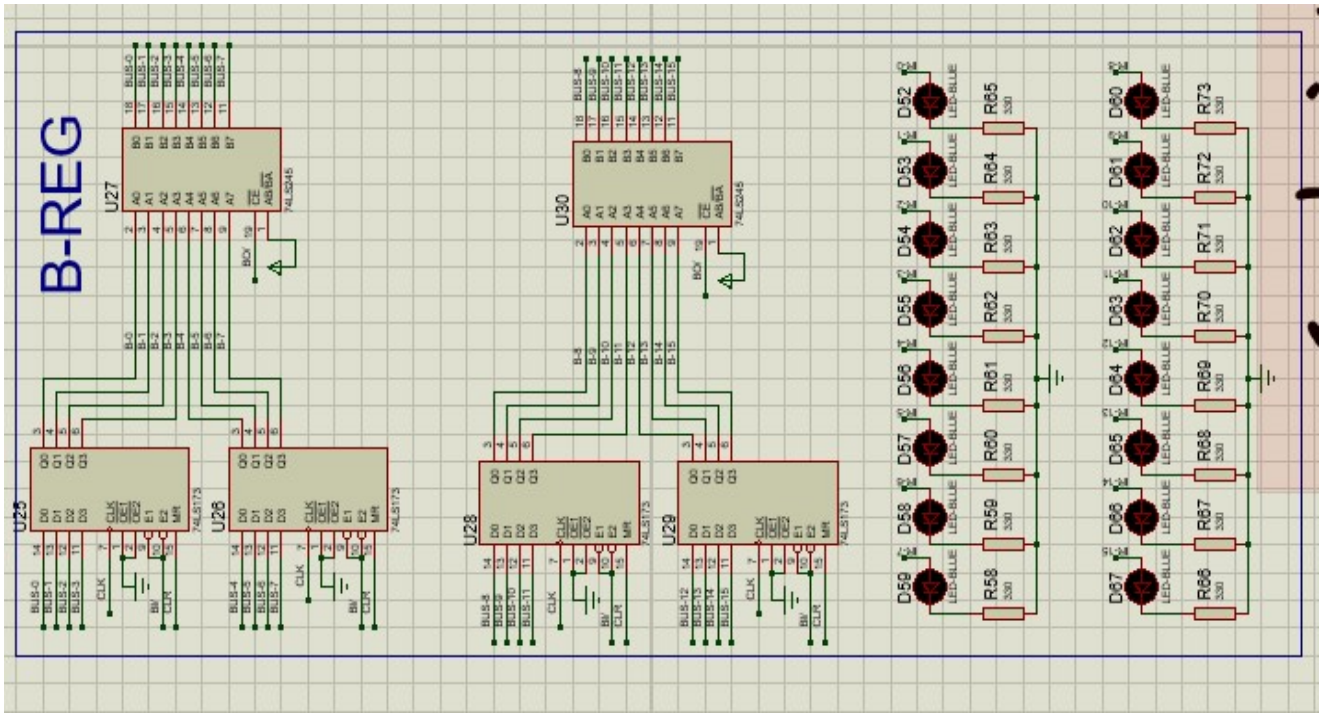


4-bits D-type Register 74173-IC



Octal Bus Transceiver 74245-IC


logic diagram (positive logic)
Output Control
Data Enable
1D
2D
CLK
3D
4D
CLR
1Q
2Q
3Q
4Q
Pin numbers shown are for D, J, N, and W packages.

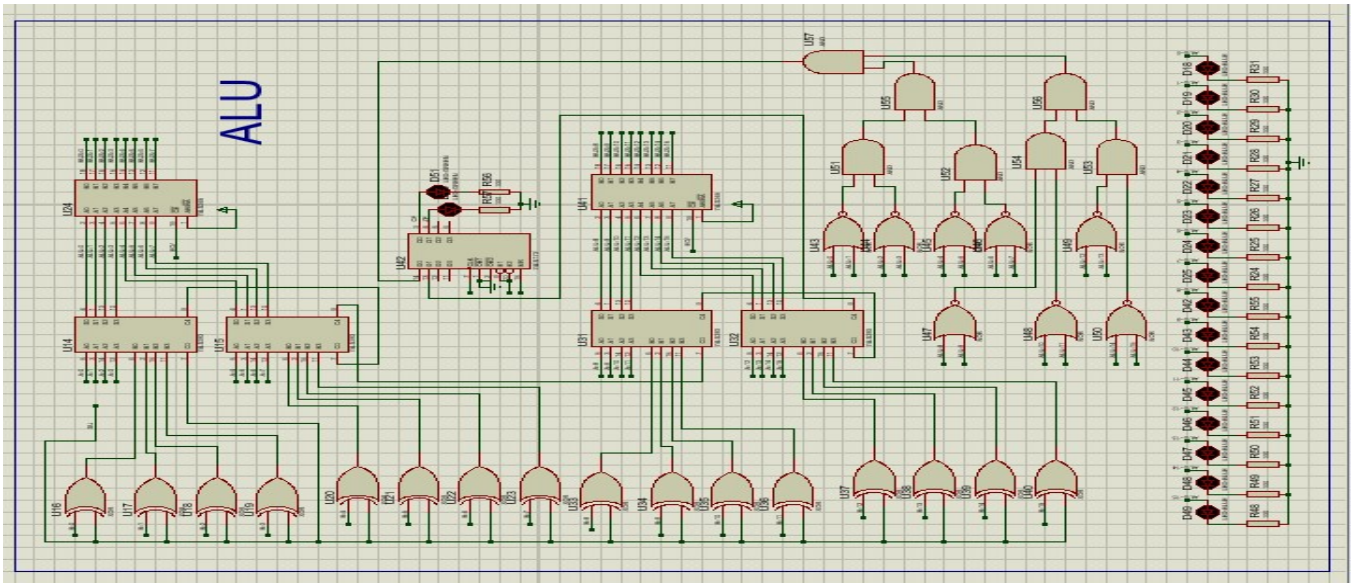

B-REG
U25
U26
U27
U28
U29
U30

Arithmetic logic unit (ALU):

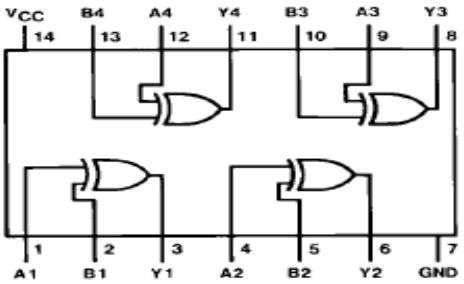
The arithmetic logic unit (ALU) part of a CPU is usually capable of performing various arithmetic, bitwise, and comparison operations on binary numbers. In our simple, the ALU is just able to add and subtract. It's connected to the A and B registers and outputs either the sum of A+B or the difference of A-B.

Using 4-Bit Binary Adder with Fast Carry (74183-IC) and connect them together to get 16 bit (Carry output to Carry input), enter b-register in XOR gate(Exclusive-OR Gate) to control add or subtract (subtract done in two's complement). Using Octal Bus Transceiver 74245-IC control to appear data on bus.

We do a carry flag because if the output of adder is be more than 16-bit send high to CPU, and do a zero flag check if the output of ALU is zero -using NOR gate - then don't complete the programming because no data. Then store data of flags in 2-bit register and connect the output of register to CPU.



Connection Diagram



Function Table

$$Y = A \oplus B = \overline{A}\,B + A\overline{B}$$

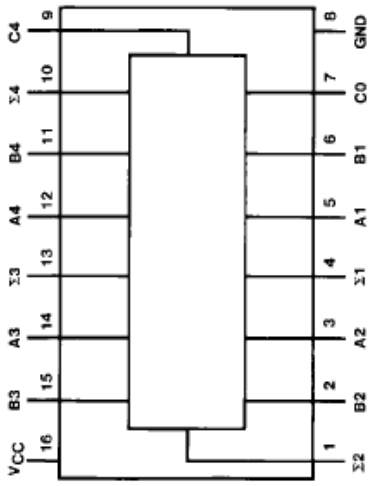
Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

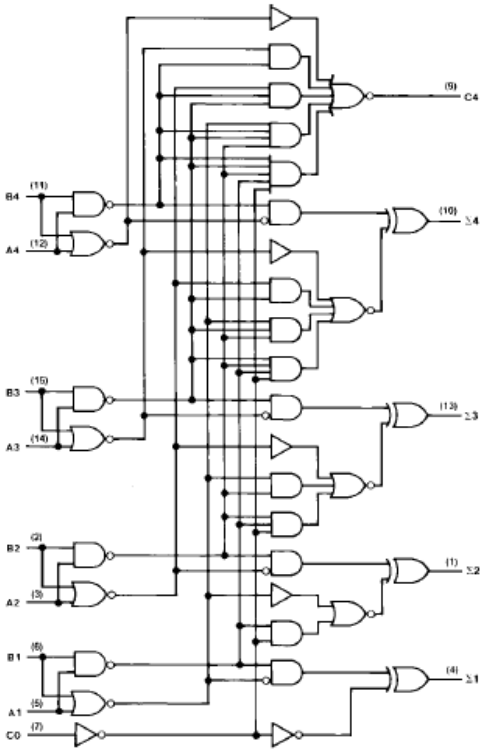
H = HIGH Logic Level
L = LOW Logic Level

Exclusive-OR Gate(XOR Gate 7486-IC)

Logic Diagram

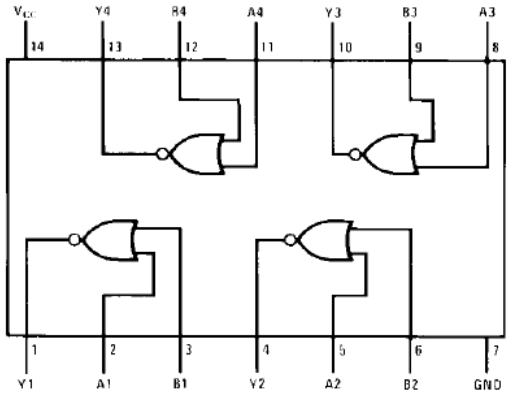
Connection Diagram





4-Bit Binary Adder with Fast Carry(74183-IC)

Connection Diagram



Function Table

$$Y = \overline{A + B}$$

Inputs		Output
A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

H = HIGH Logic Level
L = LOW Logic Level

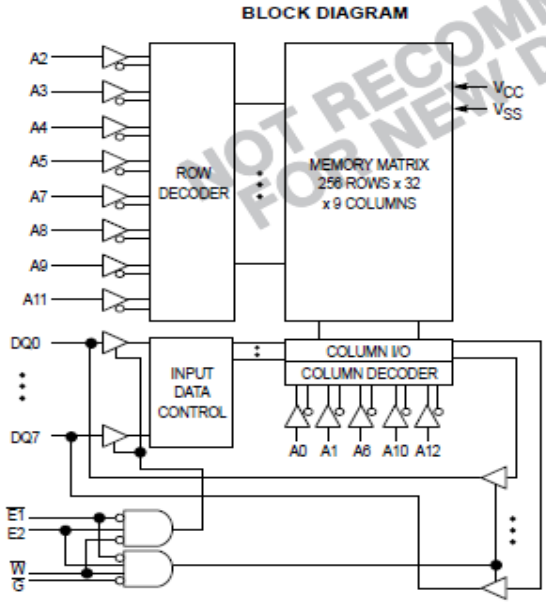
NOR Gate (7402-IC)

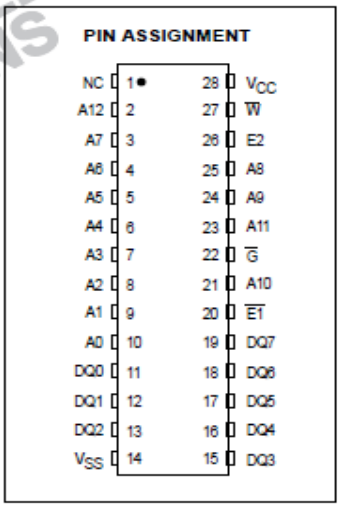
Random access memory (RAM) module :

The random access memory (RAM) stores the program the computer is executing as well as any data the program needs. Our computer uses 12-bit addresses which means it will only have 64 Kilo-bytes of RAM.

We use 6264-IC (8K x 8 Bit Fast Static RAM) we can read and write data from bus and save data in 12 address, we save address in Memory Address Register (MAR) we used 12-bit D-type register and using Quad 2-Line to 1-Line Data Selectors/Multiplexers (74175-IC) to select if we enter data and address manual from switches or from bus.

- Low Power Operation: 110 – 150 mA Maximum AC
- Fully TTL Compatible — Three State Output





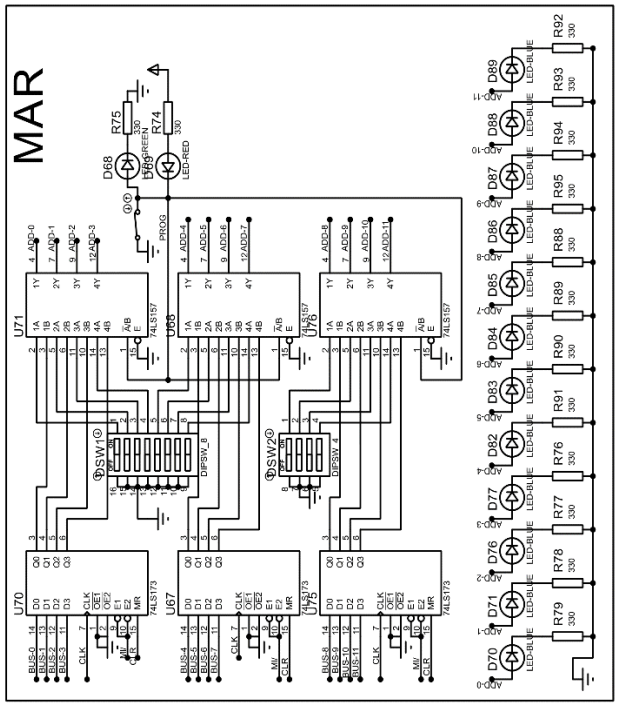
PIN NAMES

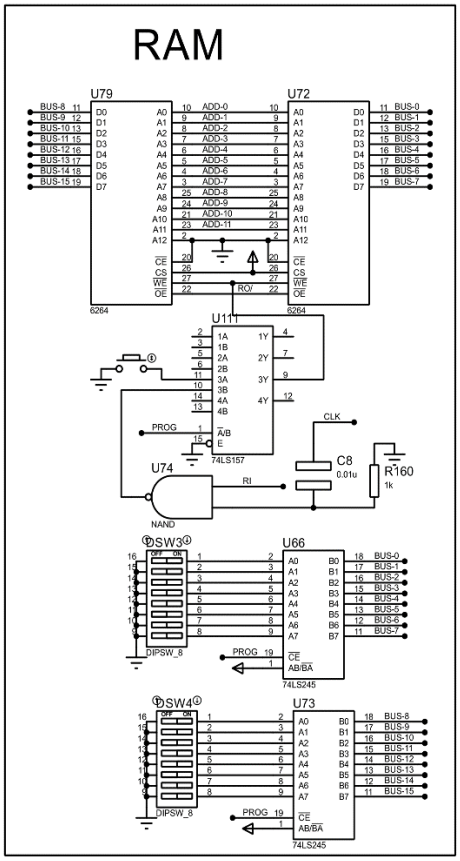
A0 – A12	Address Input
DQ0 – DQ7	Data Input/Data Output
$\overline{W}$	Write Enable
$\overline{G}$	Output Enable
$\overline{E1}$, E2	Chip Enable
V_{CC}	Power Supply (+ 5 V)
V_{SS}	Ground

TRUTH TABLE (X = Don't Care)

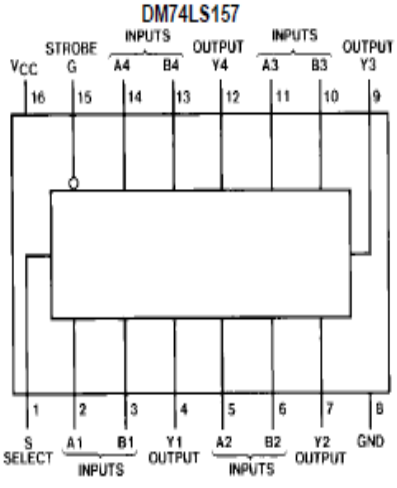
$\overline{E1}$	E2	$\overline{G}$	$\overline{W}$	Mode	V_{CC} Current	Output	Cycle
H	X	X	X	Not Selected	I_{SB1}, I_{SB2}	High–Z	—
X	L	X	X	Not Selected	I_{SB1}, I_{SB2}	High–Z	—
L	H	H	H	Output Disabled	I_{CCA}	High–Z	—
L	H	L	H	Read	I_{CCA}	D_{out}	Read Cycle
L	H	X	L	Write	I_{CCA}	High–Z	Write Cycle

6264-IC (8K x 8 Bit Fast Static RAM)

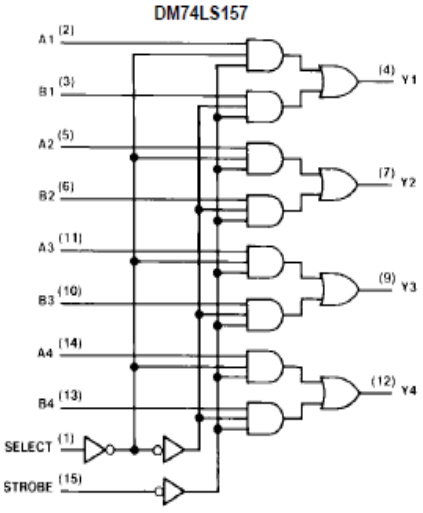




Connection Diagrams



Logic Diagrams



Function Table

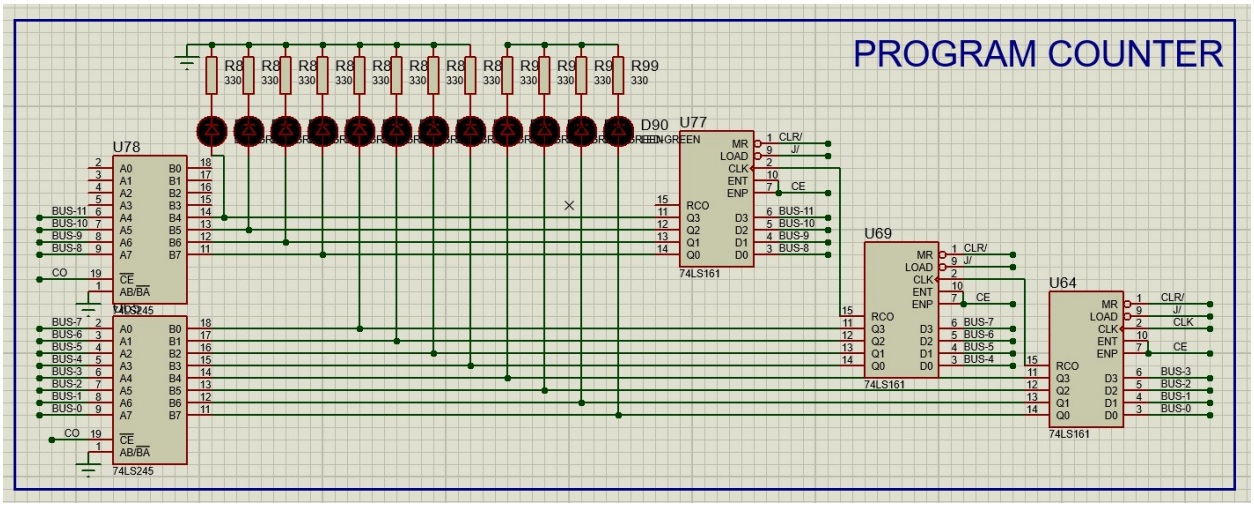
Inputs				Output Y	
Strobe	Select	A	B	DM74LS157	DM74LS158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

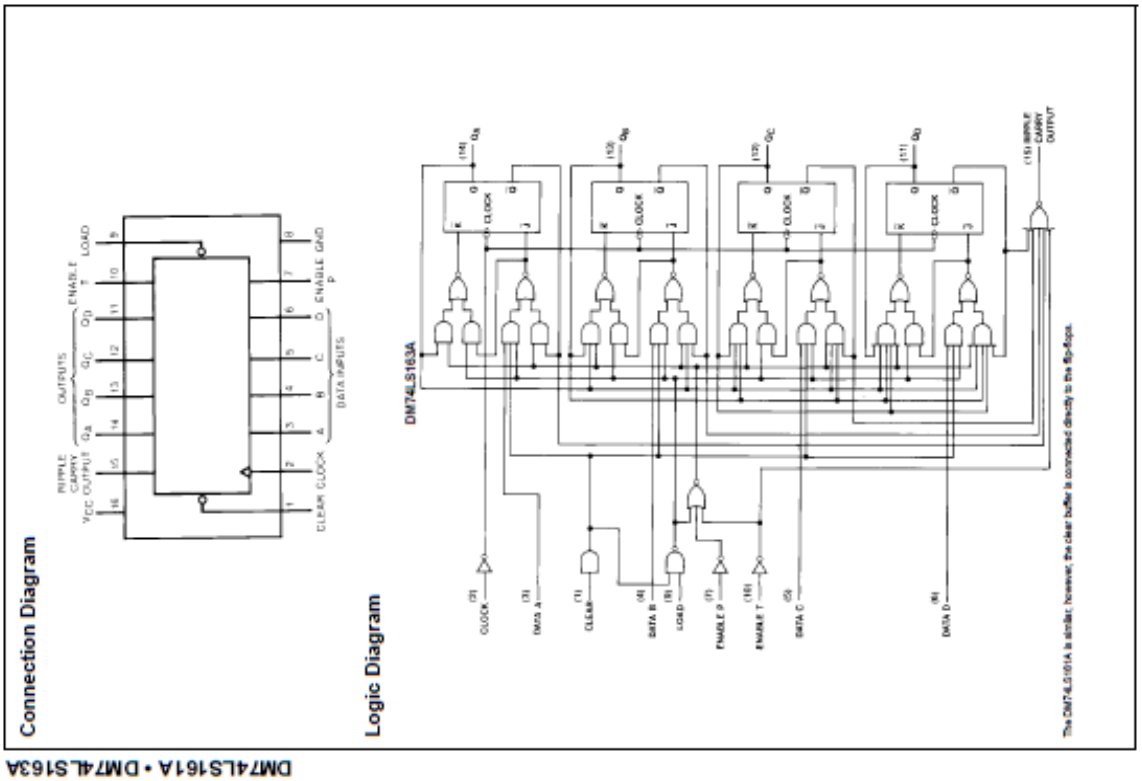
H = HIGH Level

Quad 2-Line to 1-Line Data Selectors/Multiplexers (74175-IC)

Program counter

The program counter (PC) counts in binary to keep track of which instruction the computer is currently executing. Using 4-Bit Binary Counters (74161-IC) that is have enable and rest don't like 74190-IC that have enable and not have rest or 74192-IC that don't have EN, the EN use to stop counter, we used counter 12-bit to change the address of MAD. Using the 74245-IC to contral when appear the data on bus





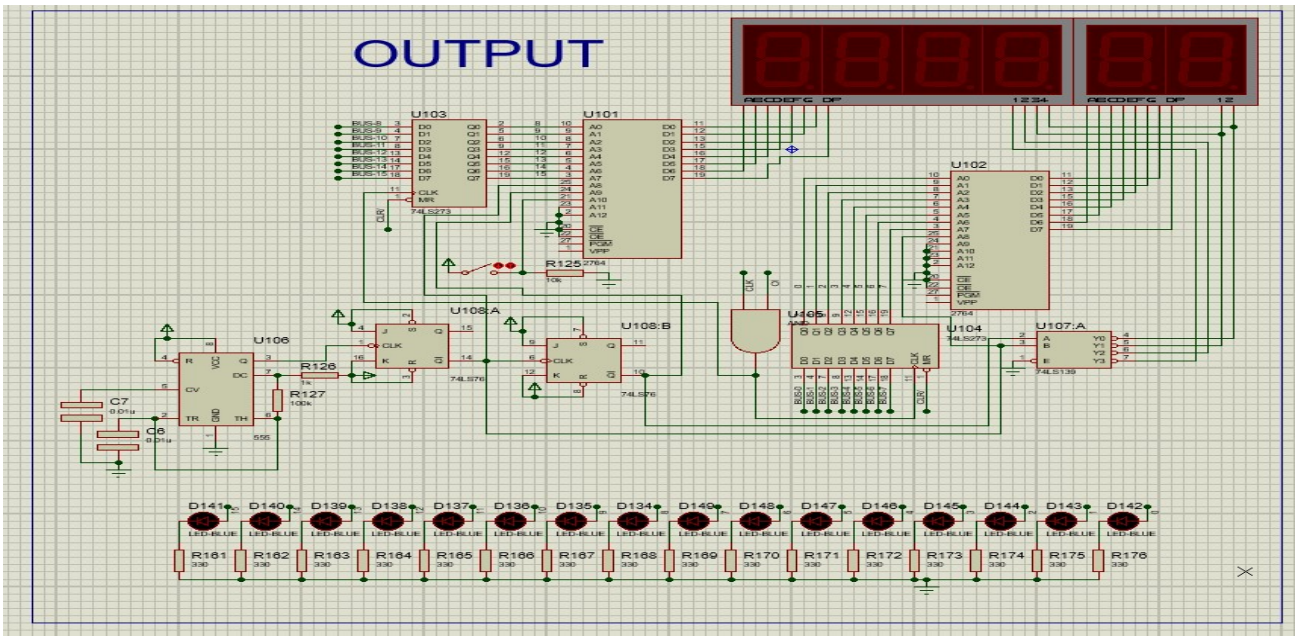
DM74LS161A • DM74LS163A

4-Bit Binary Counters

Output register

The output register is similar to any other register (like the A and B registers) except rather than displaying its contents in binary on 16 LEDs, it displays its contents in HEX on a 7-segment display (we make two way display and LEDs). Doing that requires some complex logic; luckily there's an easier way: using EEPROM.

We use 2 EEPROM first one to appear first 8-bit on 2 digit in HEX and second to appear second 2 digit and sign of number, The switch change from unsigned to sign number,. Using Dual 2-to-4 line decoder/demultiplexer(74139-IC) to select which 7-seg is read data from EEPROM and using a timer 555lm and 2 j-k flip-flop(74107-IC), as a 2 bit counter, to change the select of Demultiplexer -In high frequency all 7-seg is appear different data in same time. Using 8-bit D-type (74273-IC) to save data from bus.



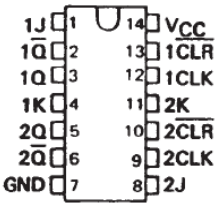
'107
FUNCTION TABLE

INPUTS				OUTPUTS	
$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	⊓	L	L	Q_0	$\overline{Q}_0$
H	⊓	H	L	H	L
H	⊓	L	H	L	H
H	⊓	H	H	TOGGLE	

'LS107A
FUNCTION TABLE

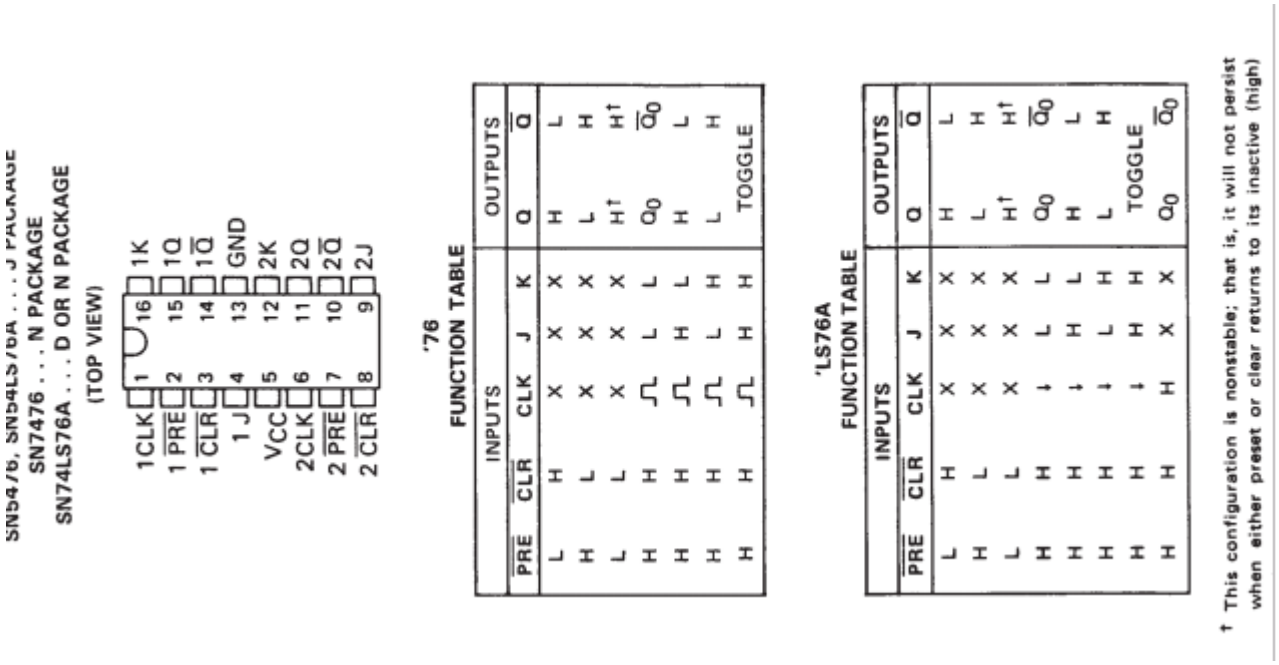
INPUTS				OUTPUTS	
$\overline{CLR}$	CLK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\overline{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\overline{Q}_0$

SN54107, SN54LS107A . . . J PACKAGE
SN74107 . . . N PACKAGE
SN74LS107A . . . D OR N PACKAGE
(TOP VIEW)



SN54LS107A . . . FK PACKAGE
(TOP VIEW)

DUAL J-K FLIP-FLOPS WITH CLEAR(74107-IC)



SN5476, SN54LS76A . . . J PACKAGE
SN7476 . . . N PACKAGE
SN74LS76A . . . D OR N PACKAGE
(TOP VIEW)

'76
FUNCTION TABLE

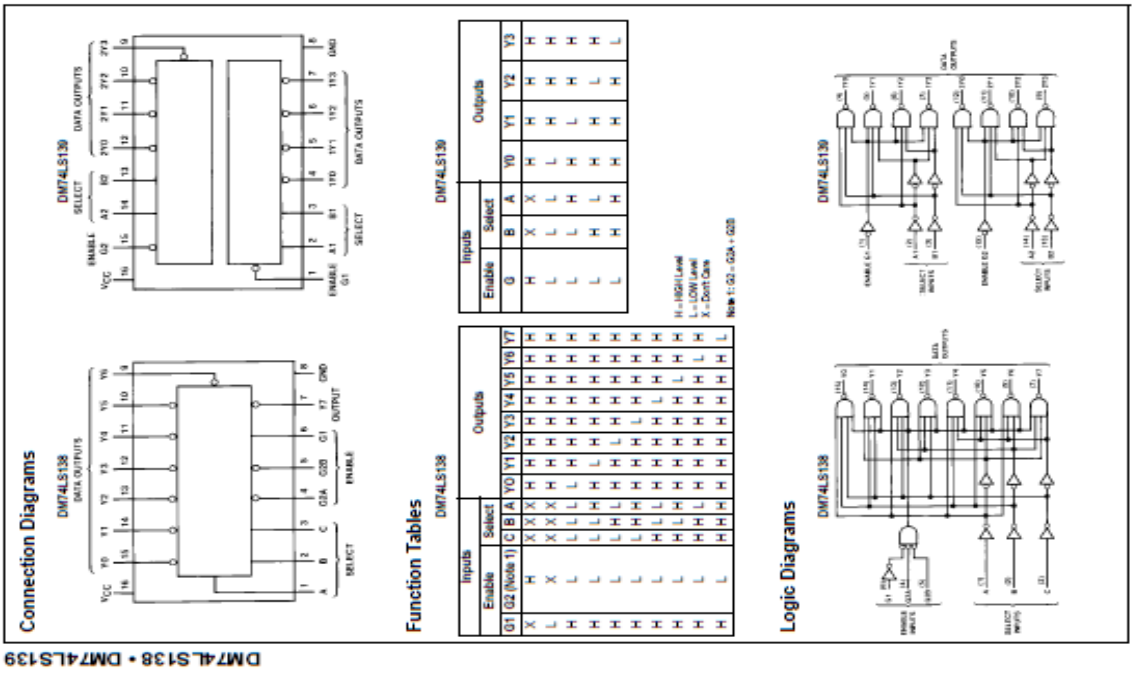
INPUTS					OUTPUTS	
PRE	CLR	CLK	J	K	Q	Q
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H†	H†
H	H	↓	L	L	Q_0	$\overline{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	

'LS76A
FUNCTION TABLE

INPUTS					OUTPUTS	
PRE	CLR	CLK	J	K	Q	Q
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H†	H†
H	H	↓	L	L	Q_0	$\overline{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\overline{Q}_0$

† This configuration is nonstable; that is, it will not persist when either preset or clear returns to its inactive (high)

DUAL J-K FLIP-FLOPS WITH CLEAR(7476-IC)



DM74LS138 • DM74LS139

Connection Diagrams

Function Tables

DM74LS138

Inputs				Outputs							
Enable		Select									
G1	G2 (Note 1)	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

DM74LS139

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

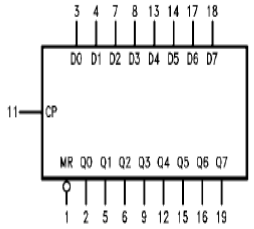
H = HIGH Level
L = LOW Level
X = Don't Care
Note 1: G2 = G2A + G2B

Logic Diagrams

www.fairchildsemi.com 2

Decoder/Demultiplexer(74138-IC&74139-IC)

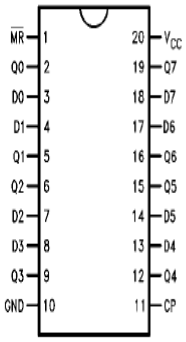
Logic Symbol



V_{CC} = Pin 20

GND = Pin 10

Connection Diagram



Pin Descriptions

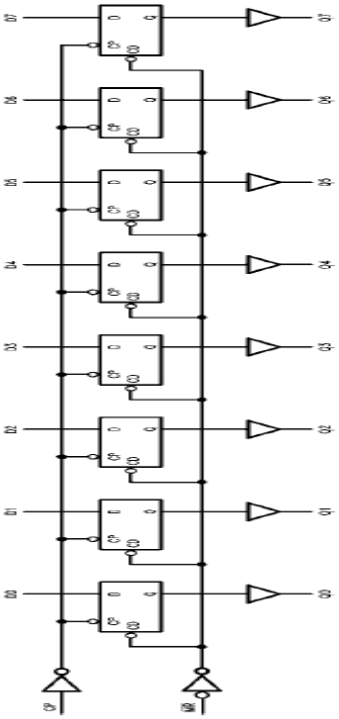
Pin Names	Description
CP	Clock Pulse Input (Active Rising Edge)
D0–D7	Data Inputs
$\overline{MR}$	Asynchronous Master Reset Input (Active LOW)
Q0–Q7	Flip-Flop Outputs

Truth Table

Inputs			Outputs
MR	CP	D_n	Q_n
L	X	X	L
H	↗	H	H
H	↗	L	L

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

Logic Diagram

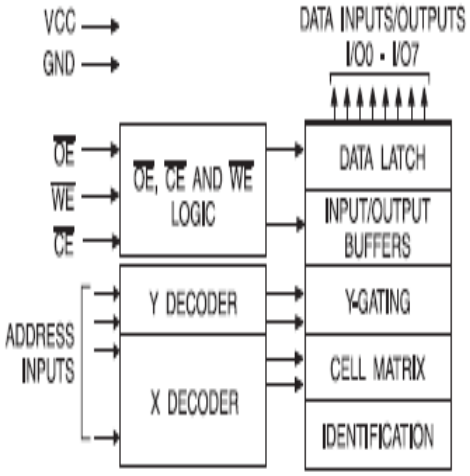


8-Bit Register with Clear(74273-IC)

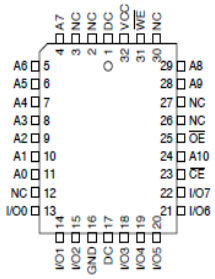
Pin Configurations

Pin Name	Function
A0 - A10	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
$\overline{WE}$	Write Enable
I/O0 - I/O7	Data Inputs/Outputs
NC	No Connect
DC	Don't Connect

Block Diagram

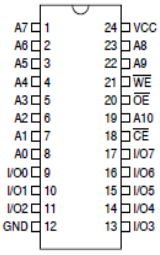


PLCC
Top View



Note: PLCC package pins 1 and 17 are DON'T CONNECT.

PDIP, SOIC
Top View

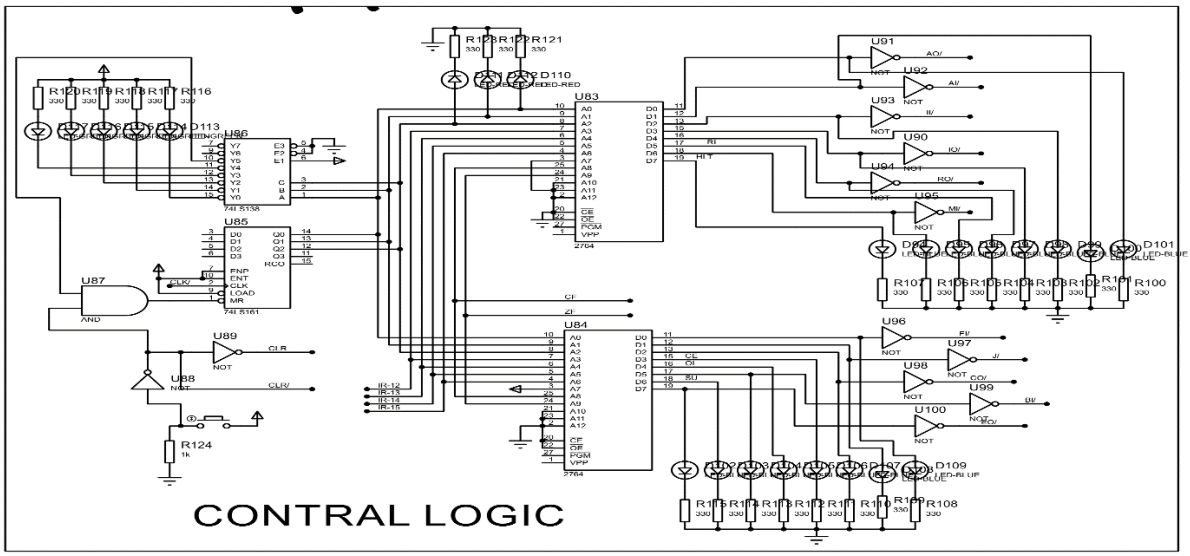


EEPROM (28c16-IC)

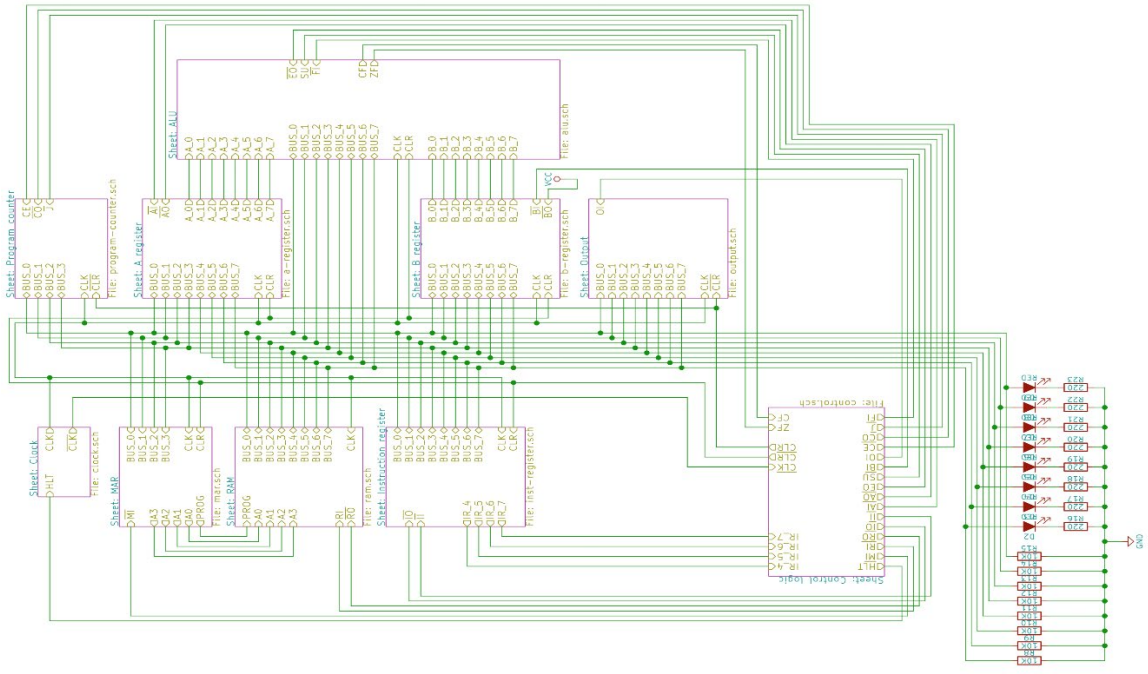
CPU control logic

The control logic is the heart of the CPU. It's what defines the opcodes the processor recognizes and what happens when it executes each instruction.

Using counter 74161-IC -to change the program to next step- and Dual 3-to-8 line decoder/demultiplexer(74138-IC) -to rest counter after 5 times- the Button to Reset all processor.



Bringing it all together



Ex:

Sum of two number :

First we put data manual in RAM by switches :

Address	*Data*
0000 0000 0000	0001 0000 0000 1110
0000 0000 0001	0010 0000 0000 1111
0000 0000 0010	1110 0000 0000 0000
0000 0000 1110	0000 0000 0001 1100 (A)
0000 0000 1111	0000 0000 0000 1110 (B)

CPU will do that before every next steps so make them as a function and name it "CPU-SET":

"CPU-SET":

1. Counter output (CO) set to appear on bus, Memory address register input (MI) set to read from the bus (counter data).
2. RAM Output (RO) set to appear the data saved in the address -which set in step (1) , the instruction register input (II) is set to read data from the bus -which in RAM- and save it , then counter enable (CE) is set to increase the counter (+1).

Second LDA :

1. "CPU-SET"
2. the instruction register Output (IO) is set and appear data in bus ,then Memory address register input (MI) set to read from the bus (instruction register) -it's read only first 12-bits.
3. RAM Output (RO) set to appear the data saved in the address -which set in last Step, the A register input (AI) is set to read data from the bus -which in RAM- and save it.

Third ADD :

1. "CPU-SET".
2. the instruction register Output (IO) is set and appear data in bus ,then Memory address register input (MI) set to read from the bus (instruction register) -it's read only first 12-bits.
3. RAM Output (RO) set to appear the data saved in the address -which set in last Step, the B register input (BI) is set to read data from the bus -which in RAM- and save it.
4. ALU Output (EO) is set to appear data on bus then , the A register input (AI) is set to read data from the bus -which in ALU- and save it.

Last OUT:

1. "CPU-SET".
2. the A register output (AO) is set and appear data in bus, then the output register input (OI) is set to save data from bus -the sum of two numbers- and appear it on display(7-seg) and 16 LEDs.

LDA		***ADD***		***OUT***	
CO				MI	
RO		II		CE	
IO	MI	IO	MI	AO	OI
RO	AI	RO	BI		
		EO	AI		