

**LIST OF EXPERIMENTS**

SL. NO.	NAME OF THE EXPERIMENT	PAGE NO.
1	Study about logic gates and verify their truth tables	1 – 5
2	Design of half adder & full adder	6 – 8
3	Design of half subtractor & full subtractor	9 – 11
4	Implement 4-bit Binary to gray code converter	12 – 14
5	Implement single bit digital comparator	15 – 16
6	Implement multiplexer and demultiplexer	17 – 20
7	Implementation of encoder and decoder	21 – 24
8	4 bit ripple counter and mod 10/mod 12 ripple counter	25 – 27
9	Design and implementation of shift register	28 – 30
10	Addition of 8-bit number	31 – 33
11	Subtraction of 8-bit number	34 – 36
12	Largest element in an array	37 – 39

**Experiment No : 01****STUDY OF LOGIC GATES**

**AIM:** To study about logic gates and verify their truth tables.

**APPARATUS REQUIRED:**

SL No.	COMPONENT	SPECIFICATION	QTY
1.	AND GATE	IC 7408	1
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
4.	NAND GATE	IC 7400	1
5.	NOR GATE	IC 7402	1
6.	X-OR GATE	IC 7486	1
7.	IC TRAINER KIT	-	1
8.	PATCH CORD	-	As per required

**THEORY:**

Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output.

OR, AND and NOT are basic gates. NAND, NOR and X-OR are known as universal gates. Basic gates form these gates.

**AND GATE:**

The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.

**OR GATE:**

The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.

**NOT GATE:**

The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.

**NAND GATE:**

The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the input is low. The output is low level when both inputs are high.

**NOR GATE:**

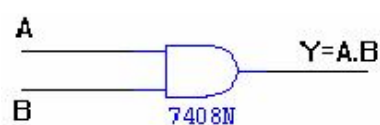
The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.

**X-OR GATE:**

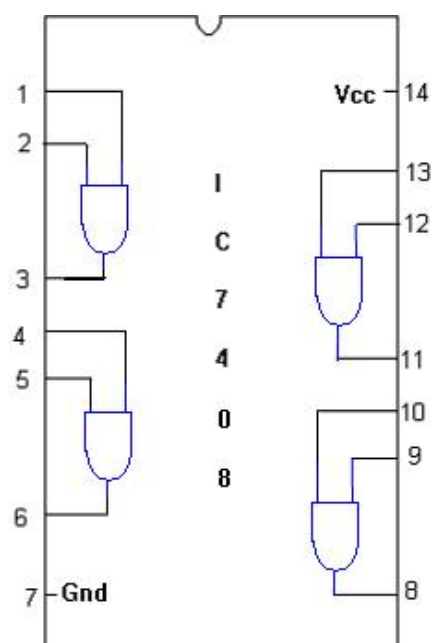
The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

**PROCEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

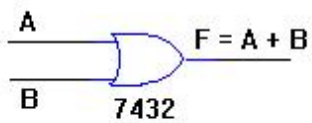
**AND GATE:****SYMBOL:****TRUTH TABLE**

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

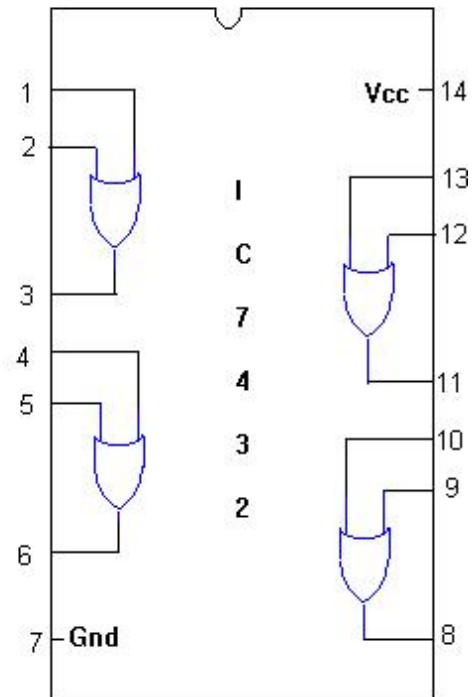
**PIN DIAGRAM:**

**OR GATE:**

SYMBOL :



PIN DIAGRAM :



TRUTH TABLE

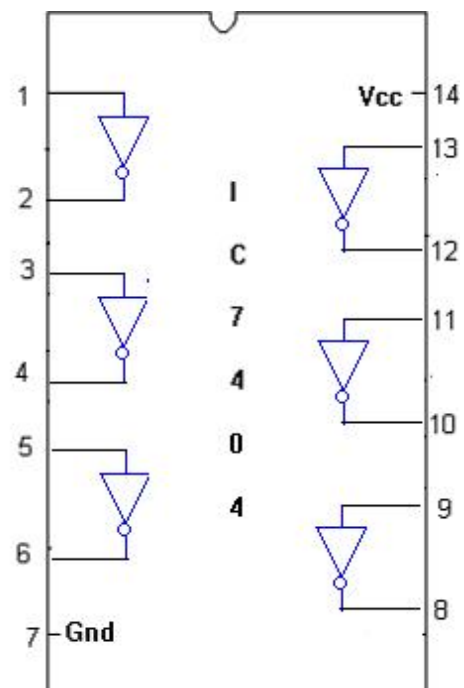
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

**NOT GATE:**

SYMBOL:



PIN DIAGRAM:

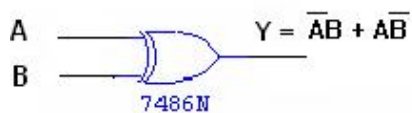


TRUTH TABLE :

A	$\bar{A}$
0	1
1	0

**X-OR GATE :**

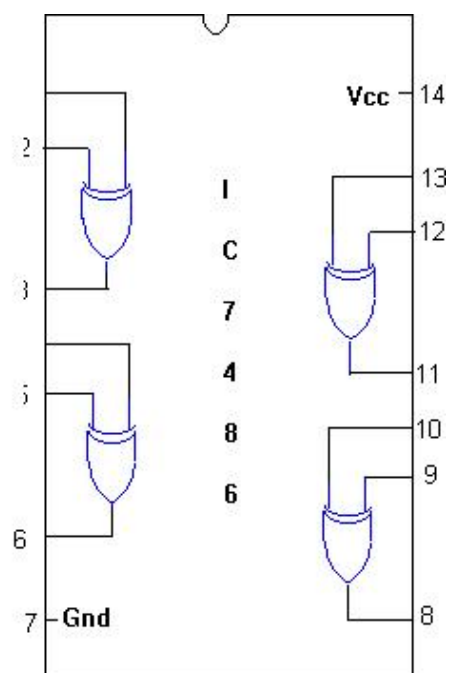
**SYMBOL :**



**TRUTH TABLE :**

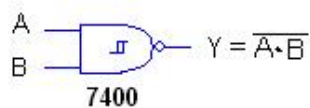
A	B	$\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

**PIN DIAGRAM :**



**NAND GATE:**

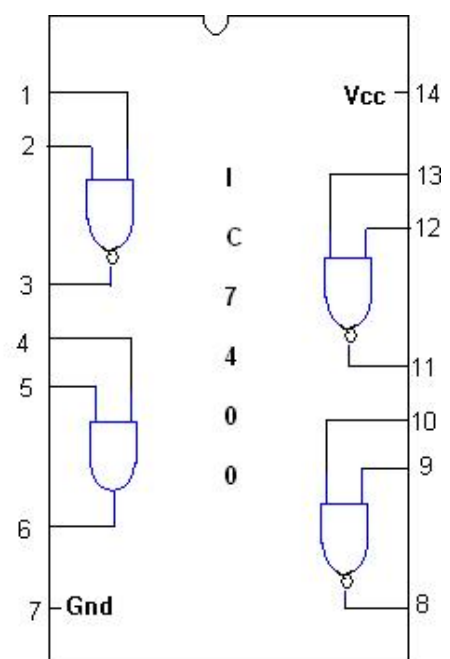
**SYMBOL:**

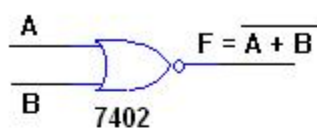


**TRUTH TABLE**

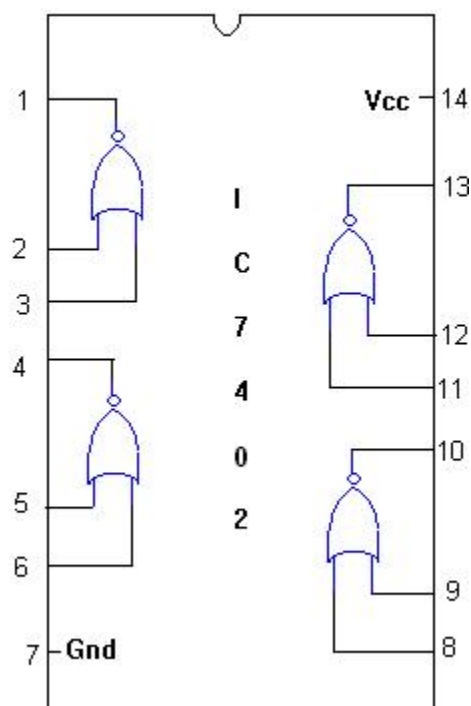
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

**PIN DIAGRAM:**



**NOR GATE:**SYMBOL :TRUTH TABLE

A	B	$\overline{A+B}$
0	0	1
0	1	1
1	0	1
1	1	0

PIN DIAGRAM :

**RESULT:** Thus the logic gates are studied and their truth tables are verified.

**Experiment No : 02****DESIGN OF HALF ADDER & FULL ADDER**

**AIM:** To design and construct half adder and full adder circuits and verify the truth table using logic gates.

**APPARATUS REQUIRED:**

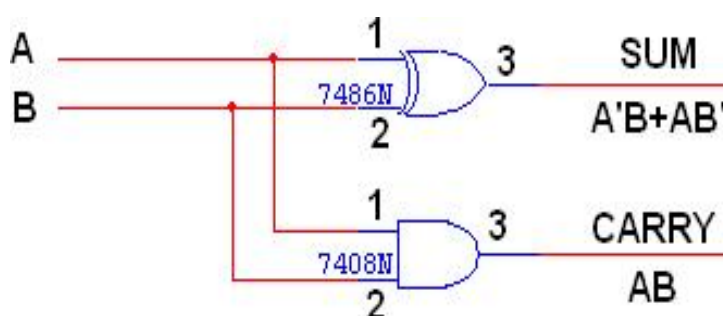
Sl. No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	X-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
4.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	As per requirement

**THEORY:****HALF ADDER:**

A half adder has two inputs for the two bits to be added and two outputs one from the sum 'S' and other from the carry 'c' into the higher adder position. Above circuit is called as a carry signal from the addition of the less significant bits sum from the X-OR Gate the carry out from the AND gate.

**FULL ADDER:**

A full adder is a combinational circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from X-OR Gate, carry output will be taken from OR Gate.

**LOGIC DIAGRAM:****HALF ADDER**

**TRUTH TABLE:**

A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

**K-Map for SUM:**

	B	00	01
A	00		1
	01	1	

$SUM = A'B + AB'$

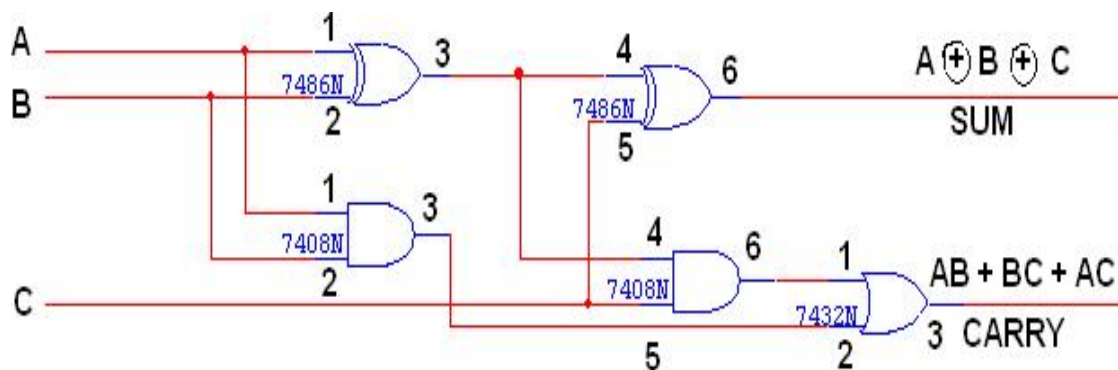
**K-Map for CARRY:**

	B	00	01
A	00		
	01		1

$CARRY = AB$

**LOGIC DIAGRAM:**

**FULL ADDER USING TWO HALF ADDER**



**TRUTH TABLE:**

A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



**K-Map for SUM:**

A \ BC	00	01	11	10
0		1		1
1	1		1	

**K-Map for CARRY:**

A \ BC	00	01	11	10
0			1	
1		1	1	1

$$\text{SUM} = A'B'C + A'BC' + ABC' + ABC$$

$$\text{CARRY} = AB + BC + AC$$

**PROCEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

**RESULT:** Thus the half adder and full adder circuits are designed and the truth tables are verified.

<b>Experiment No : 03</b>	<b>DESIGN OF HALF SUBTRACTOR &amp; FULL SUBTRACTOR</b>
---------------------------	--

**AIM:** To design and construct half subtractor and full subtractor circuits and verify the truth table using logic gates.

**APPARATUS REQUIRED:**

Sl. No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	X-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
4.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	As per requirement

**THEORY:**

**HALF SUBTRACTOR:**

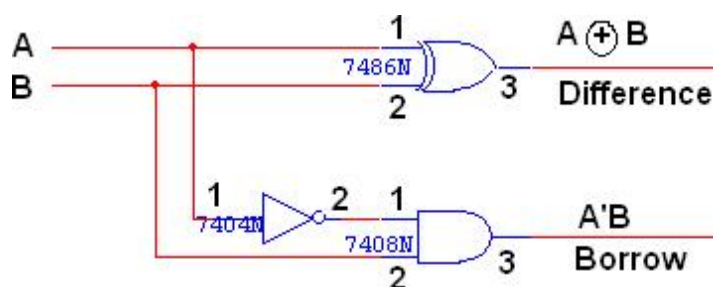
The half subtractor is constructed using X-OR and AND Gate. The half subtractor has two input and two outputs. The outputs are difference and borrow. The difference can be applied using X-OR Gate, borrow output can be implemented using an AND Gate and an inverter.

**FULL SUBTRACTOR:**

The full subtractor is a combination of X-OR, AND, OR, NOT Gates. In a full subtractor the logic circuit should have three inputs and two outputs. The two half subtractor put together gives a full subtractor. The first half subtractor will be C and A B. The output will be difference output of full subtractor. The expression  $AB$  assembles the borrow output of the half subtractor and the second term is the inverted difference output of first X-OR.

**LOGIC DIAGRAM:**

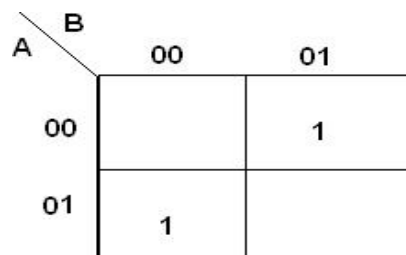
**HALF SUBTRACTOR**



**TRUTH TABLE:**

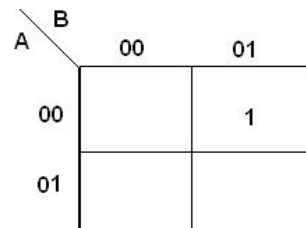
A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

**K-Map for DIFFERENCE:**



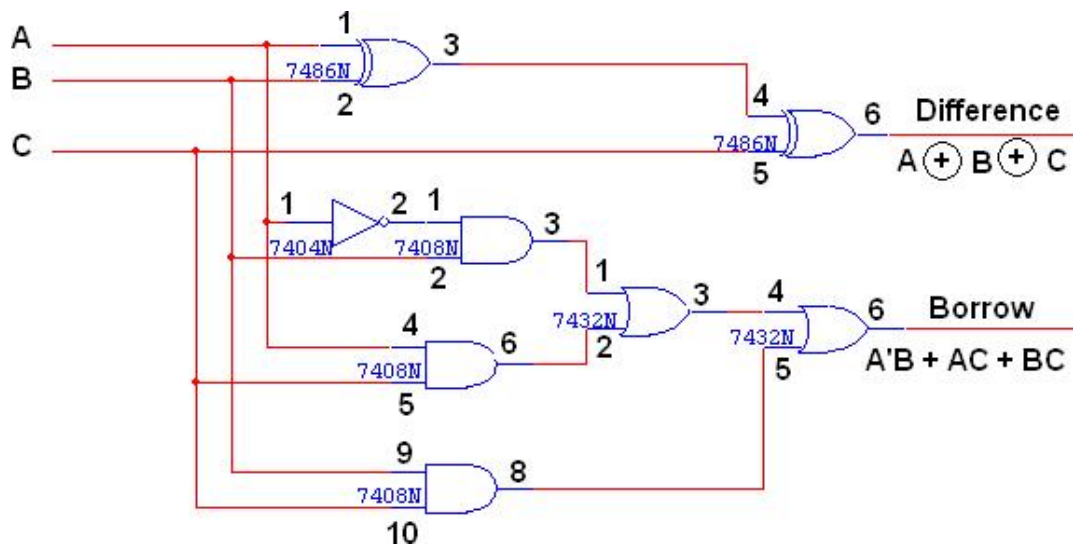
**DIFFERENCE = A'B + AB'**

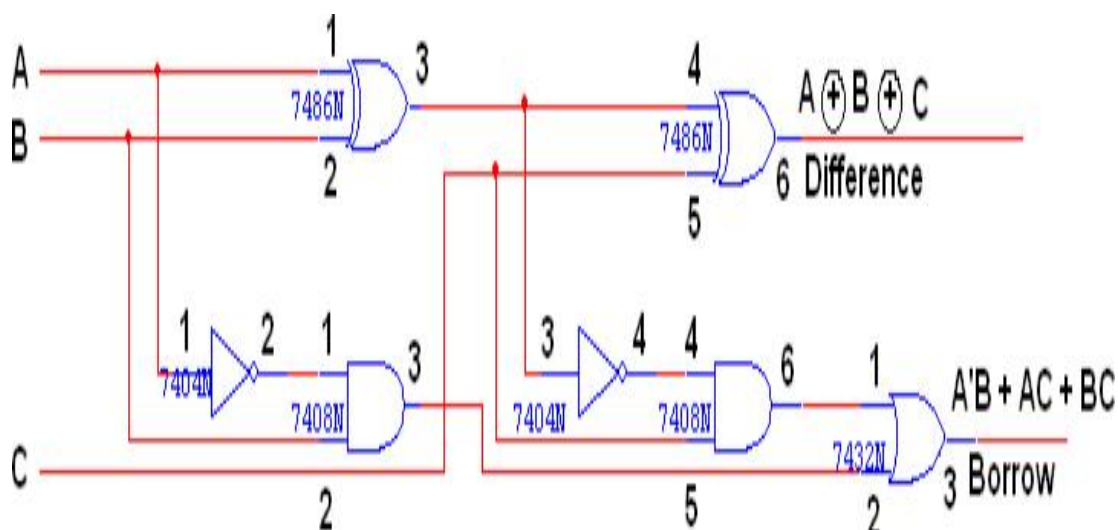
**K-Map for BORROW:**



**BORROW = A'B**

**FULL SUBTRACTOR**



**FULL SUBTRACTOR USING TWO HALF SUBTRACTOR:****TRUTH TABLE:**

A	B	C	BORROW	DIFFERENCE
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

**K-Map for Difference:**

A \ BC	00	01	11	10
0		1		1
1	1		1	

**K-Map for Borrow:**

A \ BC	00	01	11	10
0		1	1	1
1			1	

$$\text{Difference} = A'B'C + A'BC' + AB'C' + ABC$$

$$\text{Borrow} = A'B + BC + A'C$$

**PROCEEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

**RESULT:** Thus the half subtractor and full subtractor circuits are designed and the truth tables are verified.

<b>Experiment No : 04</b>	<b>BINARY TO GRAY CODE CONVERTER</b>
---------------------------	--------------------------------------

**AIM:** To design and implement 4-bit Binary to gray code converter.

**APPARATUS REQUIRED:**

Sl. No.	COMPONENT	SPECIFICATION	QTY.
1.	X-OR GATE	IC 7486	1
2.	AND GATE	IC 7408	1
3.	OR GATE	IC 7432	1
4.	NOT GATE	IC 7404	1
5.	IC TRAINER KIT	-	1
6.	PATCH CORDS	-	As per requirement

**THEORY:**

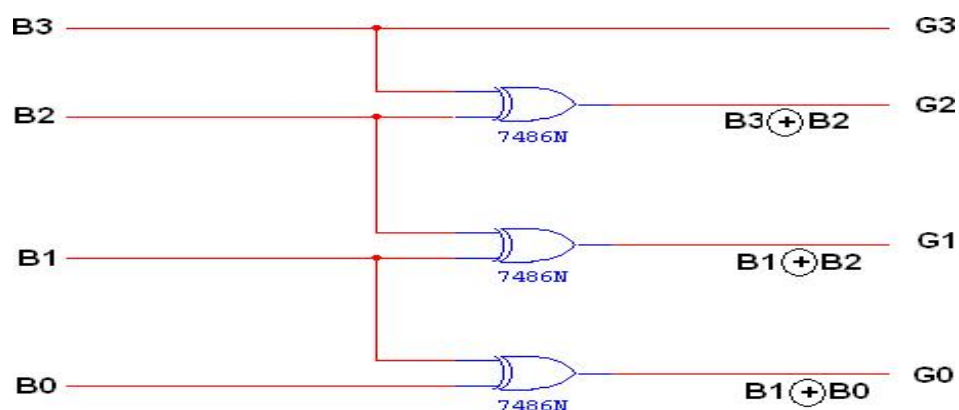
The availability of large variety of codes for the same discrete elements of information results in the use of different codes by different systems. A conversion circuit must be inserted between the two systems if each uses different codes for same information. Thus, code converter is a circuit that makes the two systems compatible even though each uses different binary code.

A code converter is a circuit that makes the two systems compatible even though each uses a different binary code. To convert from binary code to Excess-3 code, the input lines must supply the bit combination of elements as specified by code and the output lines generate the corresponding bit combination of code. Each one of the four maps represents one of the four outputs of the circuit as a function of the four input variables.

A two-level logic diagram may be obtained directly from the Boolean expressions derived by the maps. These are various other possibilities for a logic diagram that implements this circuit.

**LOGIC DIAGRAM:**

**BINARY TO GRAY CODE CONVERTOR**



**TRUTH TABLE:**

Binary input				Gray code output			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

**K-Map for G<sub>3</sub>:**

		B1B0			
		00	01	11	10
B3B2	00				
	01				
	11	1	1	1	1
	10	1	1	1	1

$G_3 = B_3$

**K-Map for G<sub>1</sub>:**

		B1B0			
		00	01	11	10
B3B2	00			1	1
	01	1	1		
	11	1	1		
	10			1	1

$G_1 = B_1 \oplus B_2$

**K-Map for G<sub>2</sub>:**

		B1B0			
		00	01	11	10
B3B2	00				
	01	1	1	1	1
	11				
	10	1	1	1	1

$G_2 = B_3 \oplus B_2$

**K-Map for G<sub>0</sub>:**

		B1B0			
		00	01	11	10
B3B2	00		1		1
	01		1		1
	11		1		1
	10		1		1

$G_0 = B_1 \oplus B_0$

**PROCEDURE:**

- (i) Connections were given as per circuit diagram.
- (ii) Logical inputs were given as per truth table
- (iii) Observe the logical output and verify with the truth tables.

**RESULT:** Thus the Binary to Gray code converter is designed and verified using truth table.

<b>Experiment No : 05</b>	<b>1-BIT MAGNITUDE COMPARATOR</b>
---------------------------	-----------------------------------

**AIM:** To design and implement single bit digital comparator.

**APPARATUS REQUIRED:**

Sl. No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	2
2.	X-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	2
4.	IC TRAINER KIT	-	1
5.	PATCH CORDS	-	As per requirement

**THEORY:**

The Digital Comparator is another very useful combinational logic circuit used to compare the value of two binary digits. A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and another for B and have three output terminals, one for  $A > B$  condition, one for  $A = B$  condition and one for  $A < B$  condition.

A comparator used to compare two bits is called a single bit comparator. It consists of two inputs each for two single-bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers

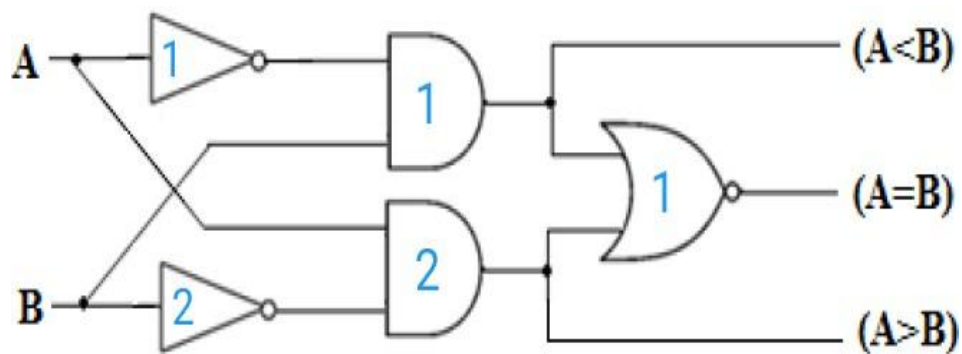
Digital comparators actually use Exclusive-NOR gates within their design for comparing their respective pairs of bits. When we are comparing two binary or BCD values or variables against each other, we are comparing the “magnitude” of these values, a logic “0” against a logic “1” which is where the term Magnitude Comparator comes from.





**TRUTH TABLE**

A	B	A > B	A = B	A < B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	1
1	1	0	1	0

**LOGIC DIAGRAM:****PROCEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

**RESULT:** Thus 1-bit magnitude comparator circuits are implemented using logic gates and IC.

**Experiment No : 06** DESIGN AND IMPLEMENTATION OF MUX AND DEMUX

**AIM:** To design and implement multiplexer and demultiplexer using logic gates and study of IC 74150 and IC 74154.

**APPARATUS REQUIRED:**

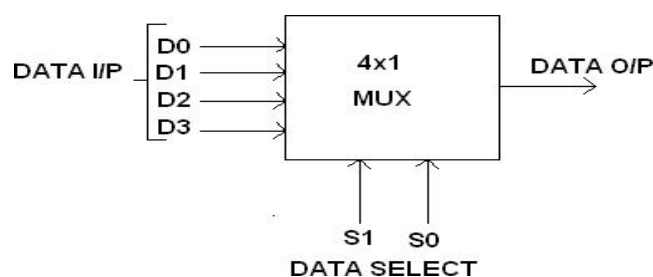
Sl. No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P AND GATE	IC 7411	2
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	32

**THEORY:****MULTIPLEXER:**

Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally there are  $2^n$  input lines and  $n$  selection lines whose bit combination determines which input is selected.

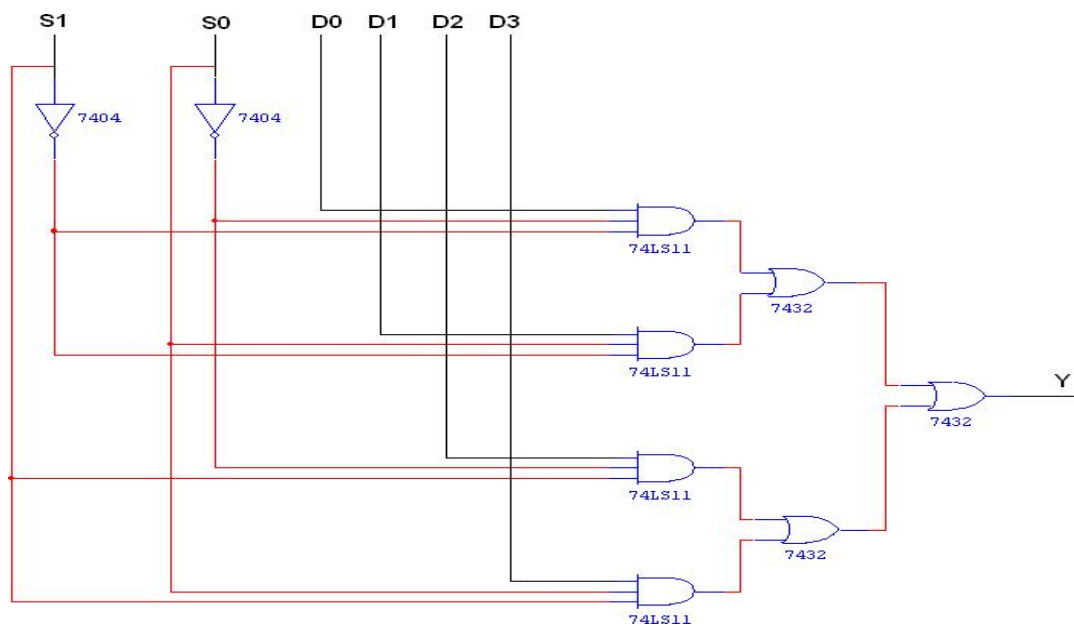
**DEMULTIPLEXER:**

The function of Demultiplexer is in contrast to multiplexer function. It takes information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoder can also be used as demultiplexer. In the 1:4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data on the data input line will pass through the selected gate to the associated data output line.

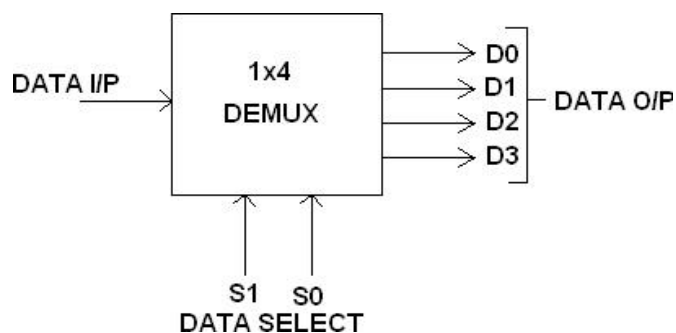
**BLOCK DIAGRAM FOR 4:1 MULTIPLEXER:****FUNCTION TABLE:**

S1	S0	OUTPUT Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

$$Y = D0 S1' S0' + D1 S1' S0 + D2 S1 S0' + D3 S1 S0$$

**CIRCUIT DIAGRAM FOR MULTIPLEXER:****TRUTH TABLE:**

S1	S0	Y = OUTPUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

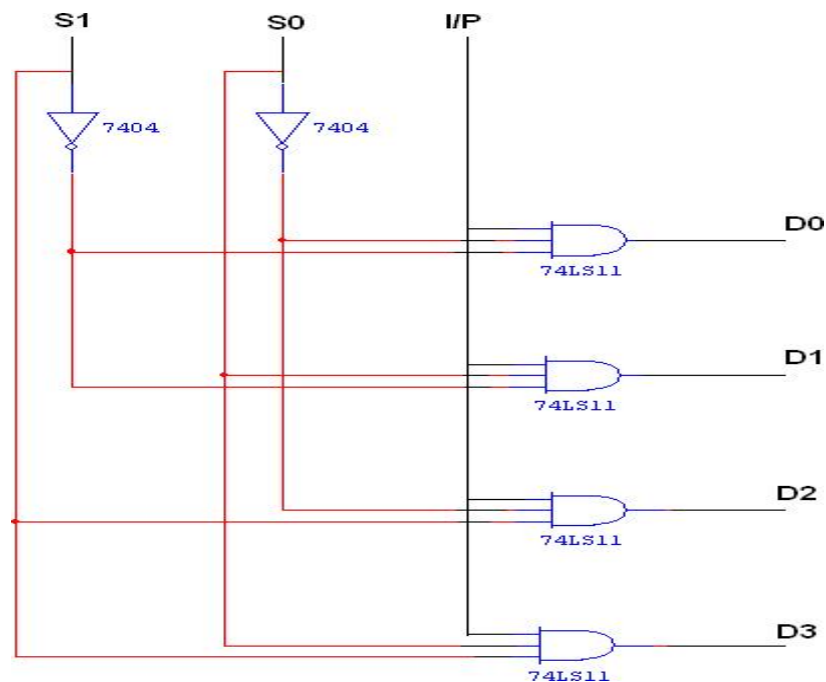
**BLOCK DIAGRAM FOR 1:4 DEMULTIPLEXER:****FUNCTION TABLE:**

S1	S0	INPUT
0	0	$X \rightarrow D0 = X S1' S0'$
0	1	$X \rightarrow D1 = X S1' S0$
1	0	$X \rightarrow D2 = X S1 S0'$
1	1	$X \rightarrow D3 = X S1 S0$

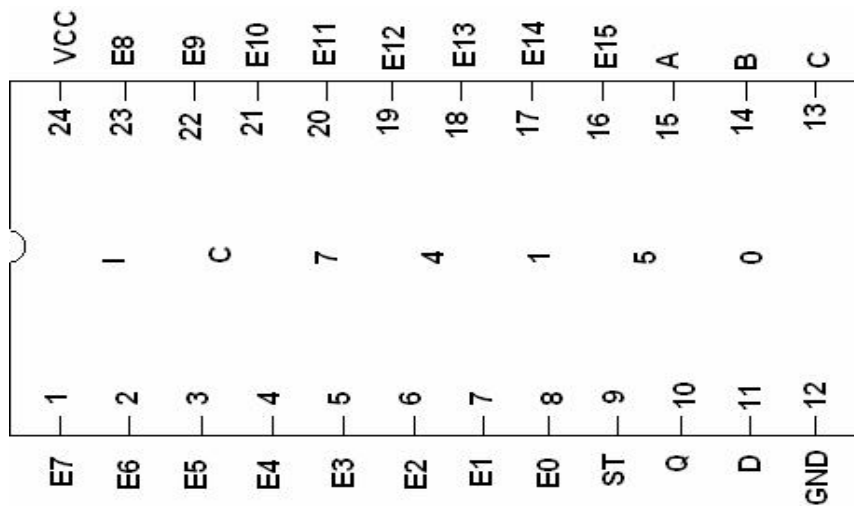
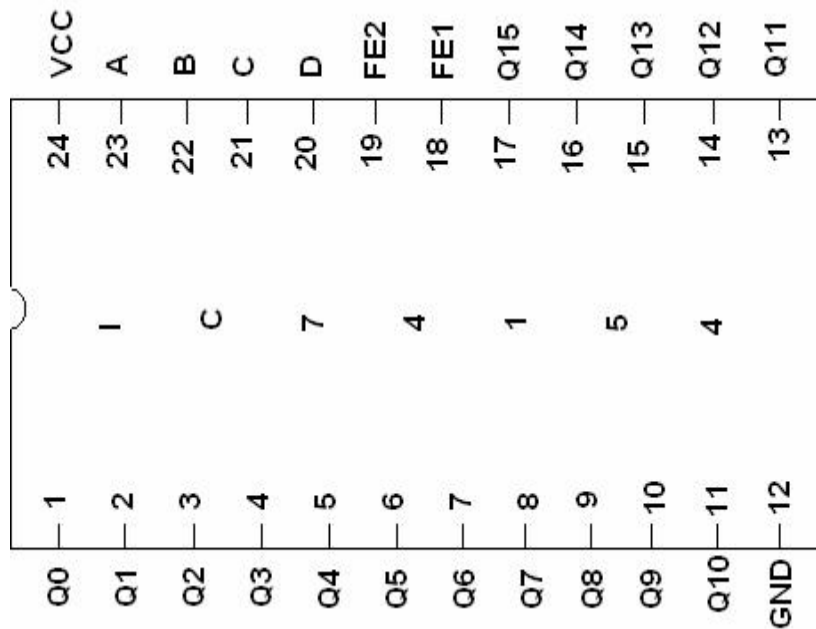
$$Y = X S1' S0' + X S1' S0 + X S1 S0' + X S1 S0$$

**TRUTH TABLE:**

INPUT			OUTPUT			
S1	S0	I/P	D0	D1	D2	D3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

**LOGIC DIAGRAM FOR DEMULTIPLEXER:****TRUTH TABLE:**

INPUT			OUTPUT			
S1	S0	I/P	D0	D1	D2	D3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

**PIN DIAGRAM FOR IC 74150:****PIN DIAGRAM FOR IC 74154:****PROCEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

**RESULT:** Thus the multiplexer and demultiplexer circuits are designed and implemented using logic gates, IC 74150 and IC 74154.

**Experiment No : 07** DESIGN AND IMPLEMENTATION OF ENCODER AND DECODER

**AIM:** To design and implement encoder and decoder using logic gates and study of IC 7445 and IC 74147.

**APPARATUS REQUIRED:**

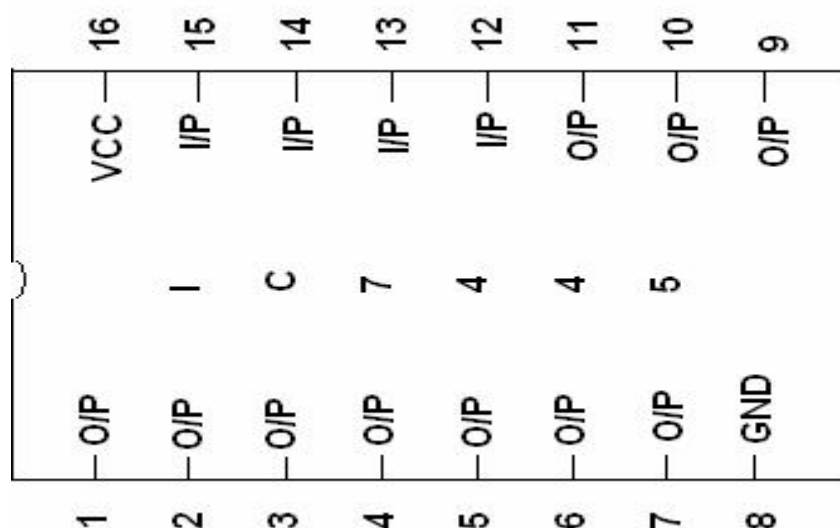
Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P NAND GATE	IC 7410	2
2.	OR GATE	IC 7432	3
3.	NOT GATE	IC 7404	1
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	27

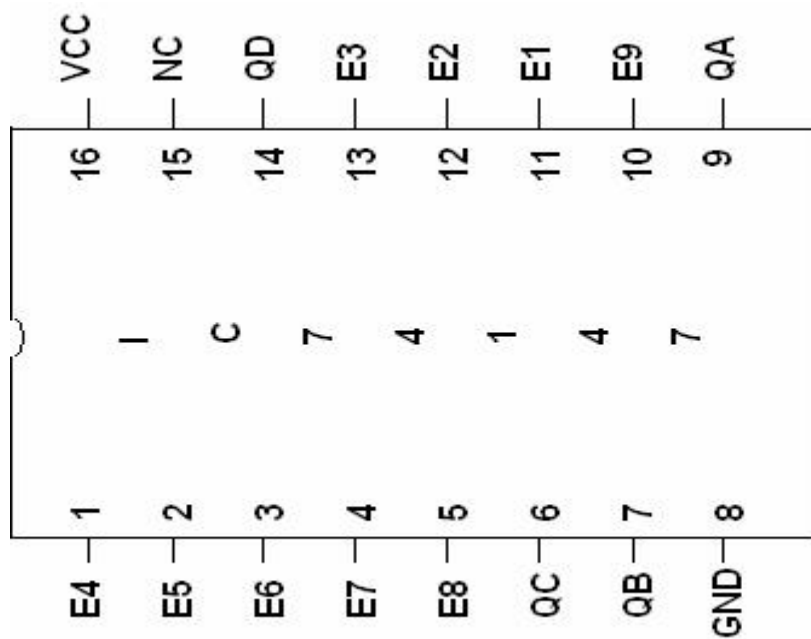
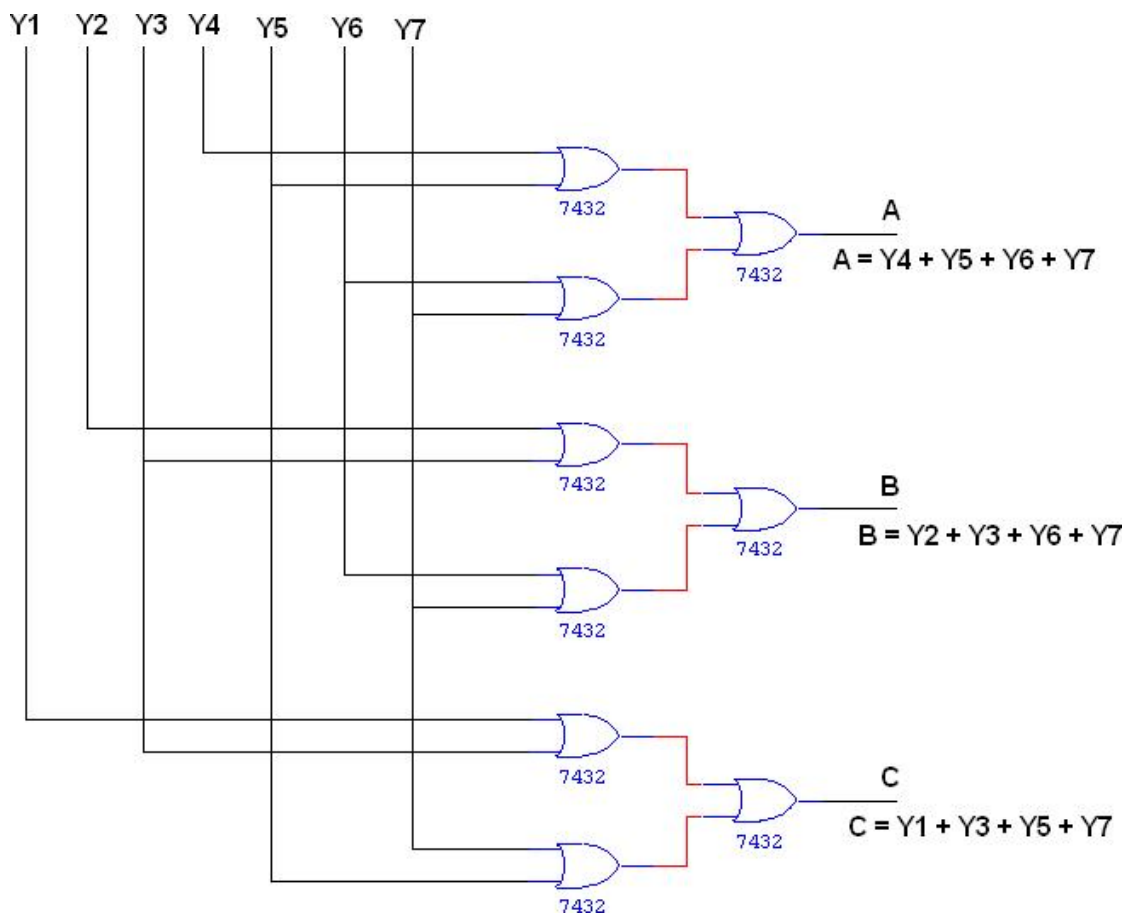
**THEORY:****ENCODER:**

An encoder is a digital circuit that perform inverse operation of a decoder. An encoder has  $2^n$  input lines and n output lines. In encoder the output lines generates the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three output that generate the corresponding binary code. In encoder it is assumed that only one input has a value of one at any given time otherwise the circuit is meaningless. It has an ambiguity that when all inputs are zero the outputs are zero. The zero outputs can also be generated when  $D_0 = 1$ .

**DECODER:**

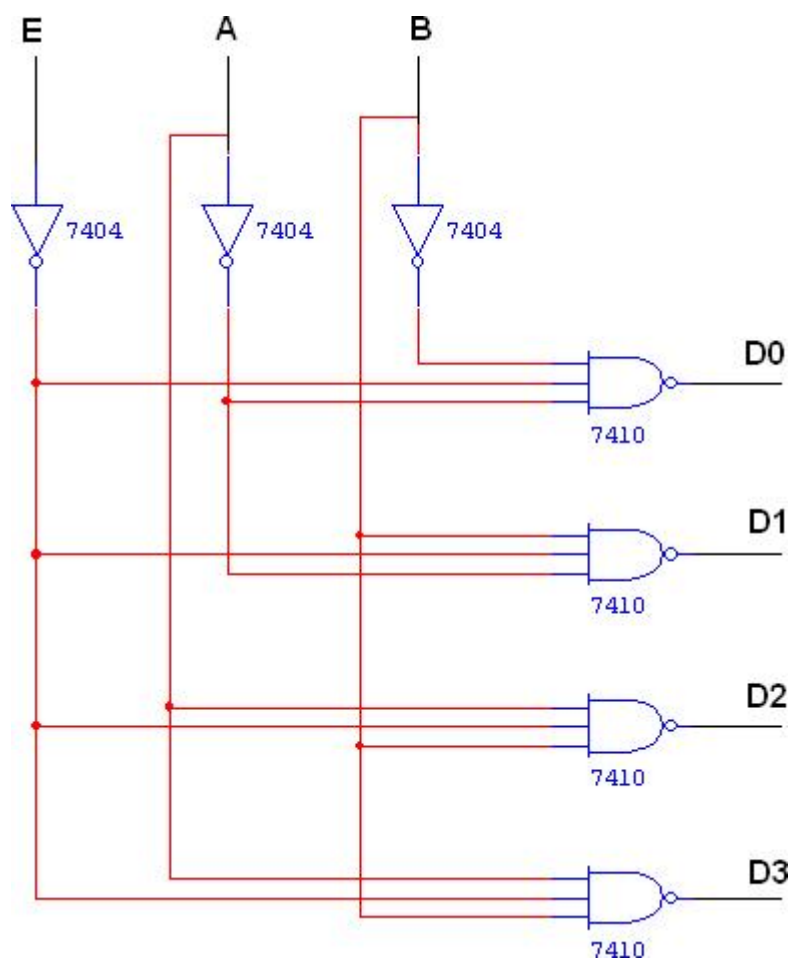
A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit the encoded information is present as n input producing  $2^n$  possible outputs.  $2^n$  output values are from 0 through out  $2^n - 1$ .

**PIN DIAGRAM FOR IC 7445:****BCD TO DECIMAL DECODER:**

**PIN DIAGRAM FOR IC 74147:****LOGIC DIAGRAM FOR ENCODER:**

**TRUTH TABLE:**

INPUT							OUTPUT		
Y1	Y2	Y3	Y4	Y5	Y6	Y7	A	B	C
1	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	1
0	0	0	1	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	1	1	1	1

**LOGIC DIAGRAM FOR DECODER:**



**TRUTH TABLE:**

INPUT			OUTPUT			
E	A	B	D0	D1	D2	D3
1	0	0	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

**PROCEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

**RESULT:** Thus the encoder and decoder circuits were designed and implemented using logicgates, IC 7445 and IC 74147.

**Experiment No : 08** 4 BIT RIPPLE COUNTER AND MOD 10/MOD 12 RIPPLE COUNTER

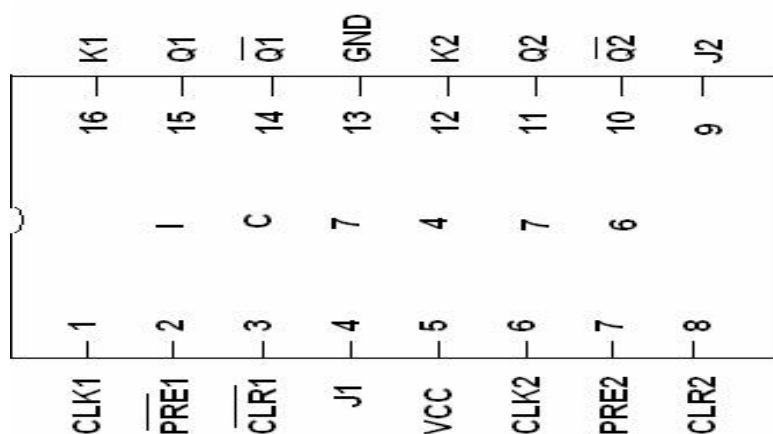
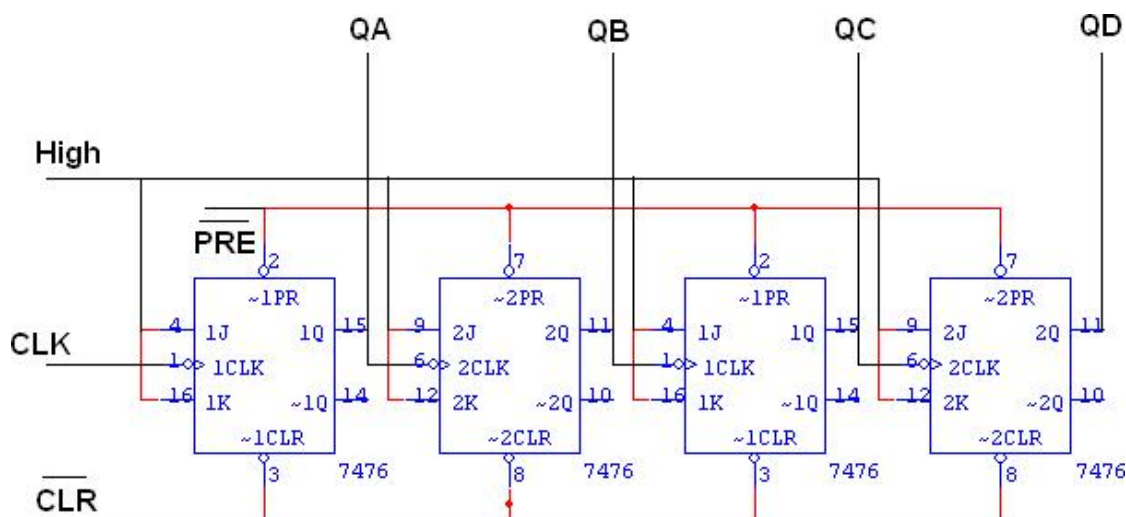
**AIM:** To design and verify 4 bit ripple counter and mod 10/ mod 12 ripple counter.

**APPARATUS REQUIRED:**

Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	JK FLIP FLOP	IC 7476	2
2.	NAND GATE	IC 7400	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	30

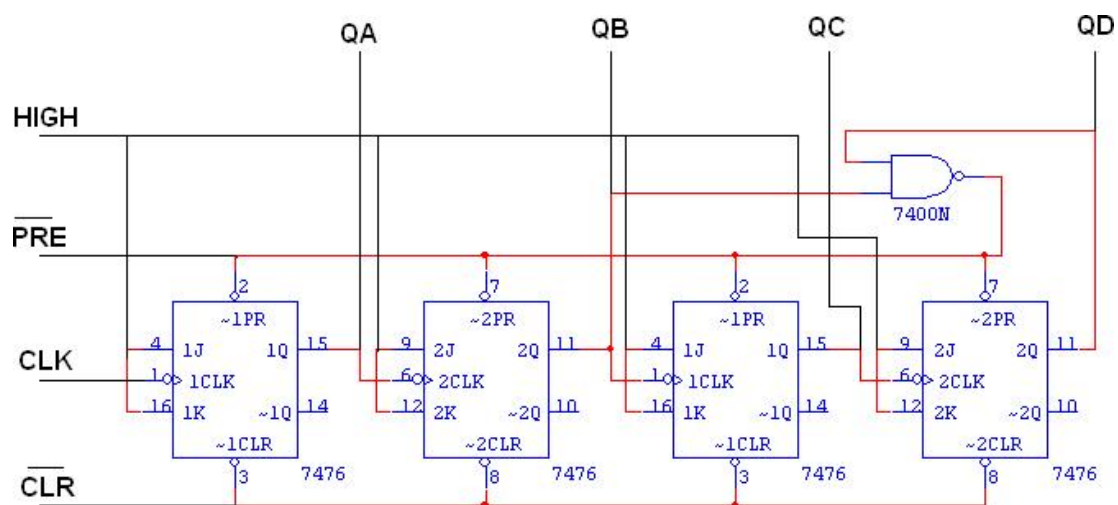
**THEORY:**

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive flip flop is clocked by Q or Q output of previous stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

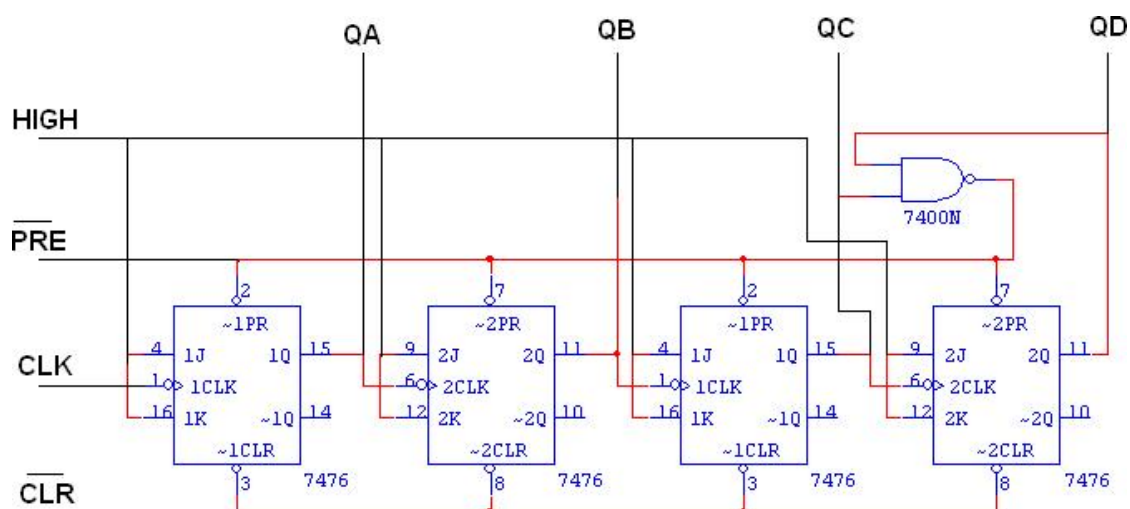
**PIN DIAGRAM FOR IC 7476:****LOGIC DIAGRAM FOR 4 BIT RIPPLE COUNTER:**

**TRUTH TABLE:**

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1

**LOGIC DIAGRAM FOR MOD - 10 RIPPLE COUNTER:****TRUTH TABLE:**

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	0	0	0

**LOGIC DIAGRAM FOR MOD - 12 RIPPLE COUNTER:****TRUTH TABLE:**

CLK	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	0	0

**PROCEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

**RESULT:** Thus the 4-bit ripple counter mod 10/ mod 12 ripple counter circuits were designed and verified successfully.

<b>Experiment No : 09</b>	<b>DESIGN AND IMPLEMENTATION OF SHIFT REGISTER</b>
---------------------------	--

**AIM:** To design and implement Serial in serial out , Serial in parallel out , Parallel in serial out and Parallel in parallel out shift register.

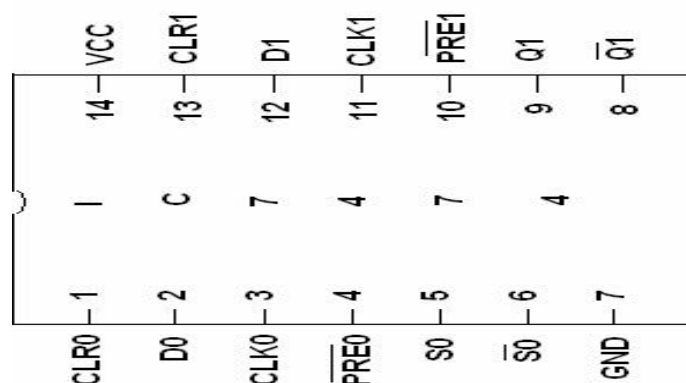
**APPARATUS REQUIRED:**

Sl. No.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	35

**THEORY:**

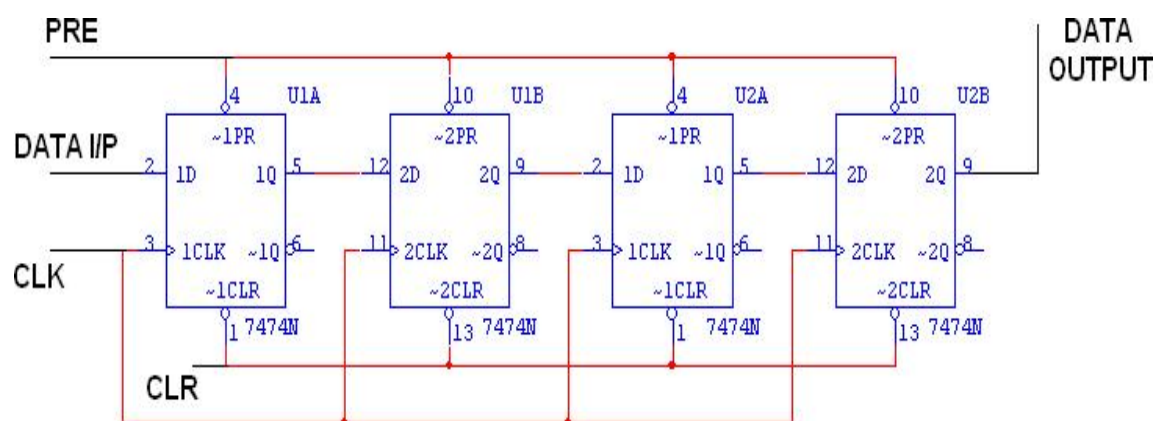
A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only flip flop.

**PIN DIAGRAM:**



**LOGIC DIAGRAM:**

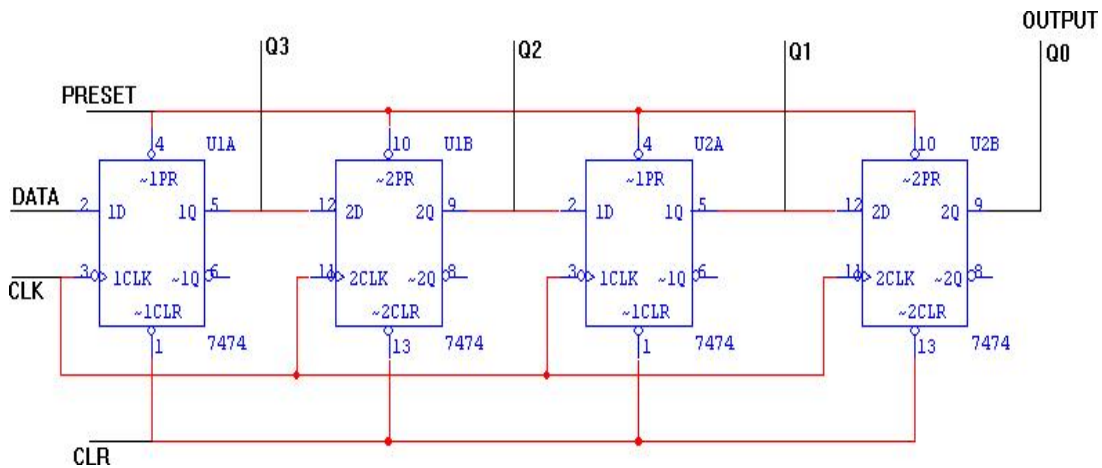
**SERIAL IN SERIAL OUT:**



**TRUTH TABLE:**

CLK	Serial in	Serial out
1	1	0
2	0	0
3	0	0
4	1	1
5	X	0
6	X	0
7	X	1

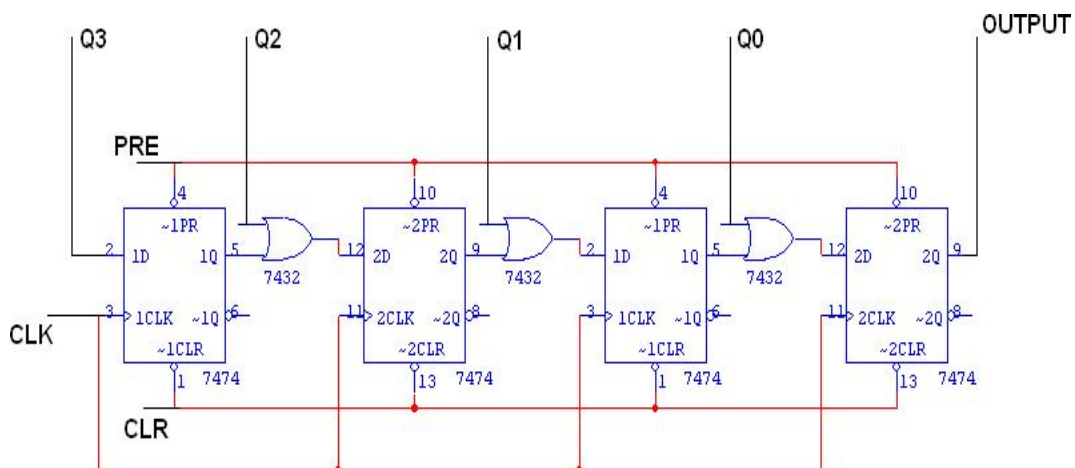
**LOGIC DIAGRAM:  
SERIAL IN PARALLEL OUT:**



**TRUTH TABLE:**

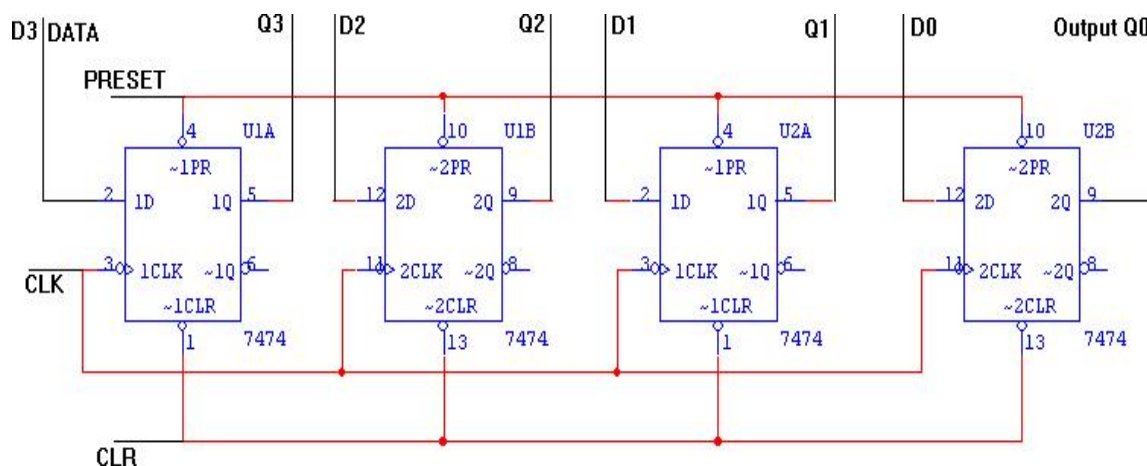
CLK	DATA	OUTPUT			
		QA	QB	QC	QD
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	0	0	1

**LOGIC DIAGRAM:  
PARALLEL IN SERIAL OUT:**



**TRUTH TABLE:**

CLK	Q3	Q2	Q1	Q0	O/P
0	1	0	0	1	1
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	1

**LOGIC DIAGRAM:****PARALLEL IN PARALLEL OUT:****TRUTH TABLE:**

CLK	DATA INPUT				OUTPUT			
	D <sub>A</sub>	D <sub>B</sub>	D <sub>C</sub>	D <sub>D</sub>	Q <sub>A</sub>	Q <sub>B</sub>	Q <sub>C</sub>	Q <sub>D</sub>
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

**PROCEDURE:**

- (i) Connections are given as per circuit diagram.
- (ii) Logical inputs are given as per circuit diagram.
- (iii) Observe the output and verify the truth table.

**RESULT:** Thus the shift registers were designed and implemented using IC7474 and verified successfully.

**Experiment No : 10****ADDITION OF 8-BIT NUMBER**

**AIM:** To add two 8 bit numbers stored at consecutive memory locations.

**ALGORITHM:**

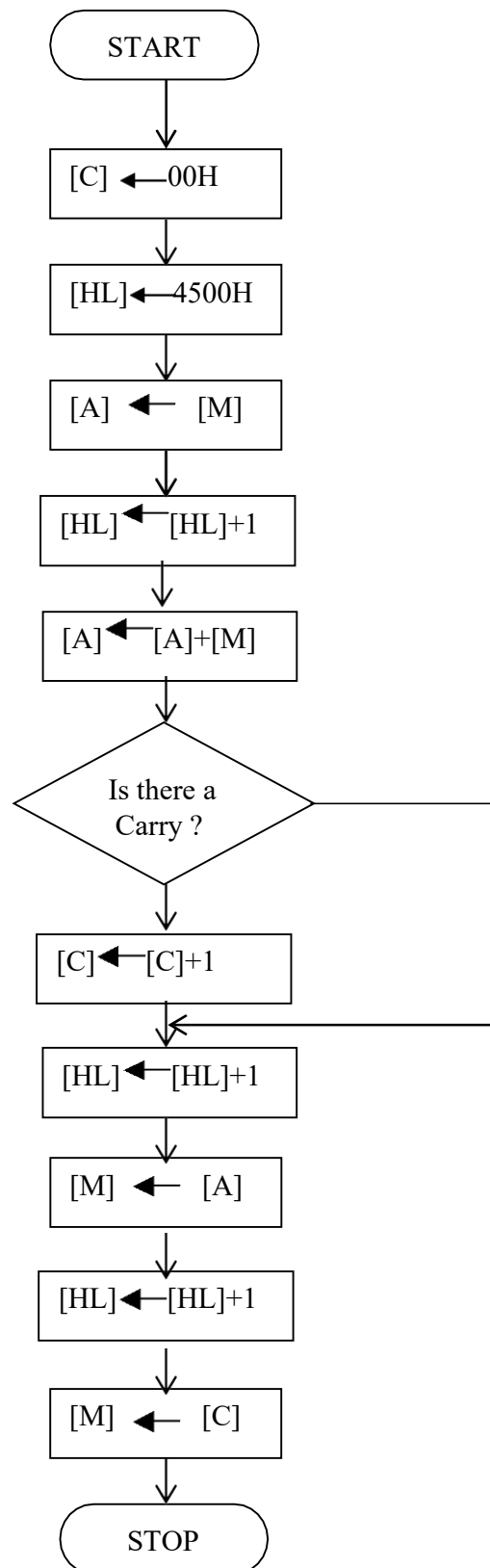
1. Initialize memory pointer to data location.
2. Get the first number from memory in accumulator.
3. Get the second number and add it to the accumulator.
4. Store the answer at another memory location.

**RESULT:**

Thus the 8 bit numbers stored at 4500 & 4501 are added and the result stored at 4502 & 4503.



## FLOW CHART:



**PROGRAM:**

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100		START	MVI	C, 00	Clear C reg.
4101					
4102			LXI	H, 4500	Initialize HL reg. to 4500
4103					
4104					
4105			MOV	A, M	Transfer first data to accumulator
4106			INX	H	Increment HL reg. to point next memory Location.
4107			ADD	M	Add first number to acc. Content.
4108			JNC	L1	Jump to location if result does not yield carry.
4109					
410A					
410B			INR	C	Increment C reg.
410C		L1	INX	H	Increment HL reg. to point next memory Location.
410D			MOV	M, A	Transfer the result from acc. to memory.
410E			INX	H	Increment HL reg. to point next memory Location.
410F			MOV	M, C	Move carry to memory
4110			HLT		Stop the program

**OBSERVATION:**

INPUT		OUTPUT	
4500		4502	
4501		4503	

**Experiment No : 11****SUBTRACTION OF 8-BIT NUMBER**

**AIM:** To Subtract two 8 bit numbers stored at consecutive memory locations.

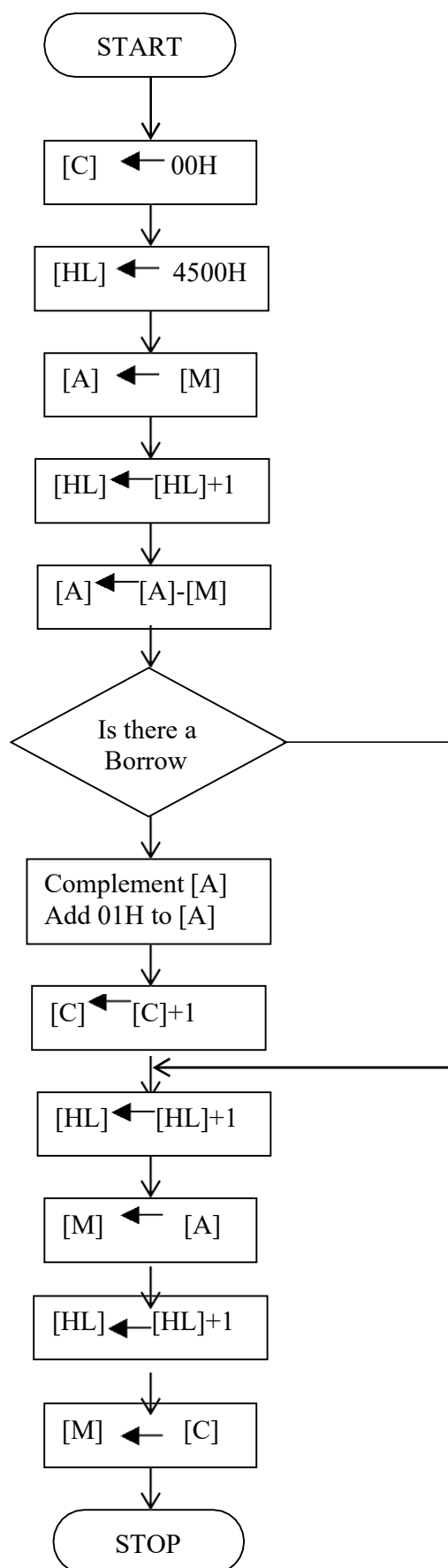
**ALGORITHM:**

1. Initialize memory pointer to data location.
2. Get the first number from memory in accumulator.
3. Get the second number and subtract from the accumulator.
4. If the result yields a borrow, the content of the acc. is complemented and 01H is added to it (2's complement). A register is cleared and the content of that reg. is incremented in case there is a borrow. If there is no borrow the content of the acc. is directly taken as the result.
5. Store the answer at next memory location.

**RESULT:**

Thus the 8 bit numbers stored at 4500 & 4501 are subtracted and the result stored at 4502 & 4503.

## FLOW CHART:



**PROGRAM:**

ADDRESS	OPCODE	LABEL	MNEMONICS	OPERAND	COMMENT
4100		START	MVI	C, 00	Clear C reg.
4101					
4102			LXI	H, 4500	Initialize HL reg. to 4500
4103					
4104					
4105			MOV	A, M	Transfer first data to accumulator
4106			INX	H	Increment HL reg. to point next mem. Location.
4107			SUB	M	Subtract first number from acc. Content.
4108			JNC	L1	Jump to location if result does not yield borrow.
4109					
410A					
410B			INR	C	Increment C reg.
410C			CMA		Complement the Acc. content
410D			ADI	01H	Add 01H to content of acc.
410E					
410F		L1	INX	H	Increment HL reg. to point next mem. Location.
4110			MOV	M, A	Transfer the result from acc. to memory.
4111			INX	H	Increment HL reg. to point next mem. Location.
4112			MOV	M, C	Move carry to mem.
4113			HLT		Stop the program

**OBSERVATION:**

INPUT		OUTPUT	
4500		4502	
4501		4503	

**Experiment No : 12****LARGEST ELEMENT IN AN ARRAY**

**AIM:** To find the largest element in an array.

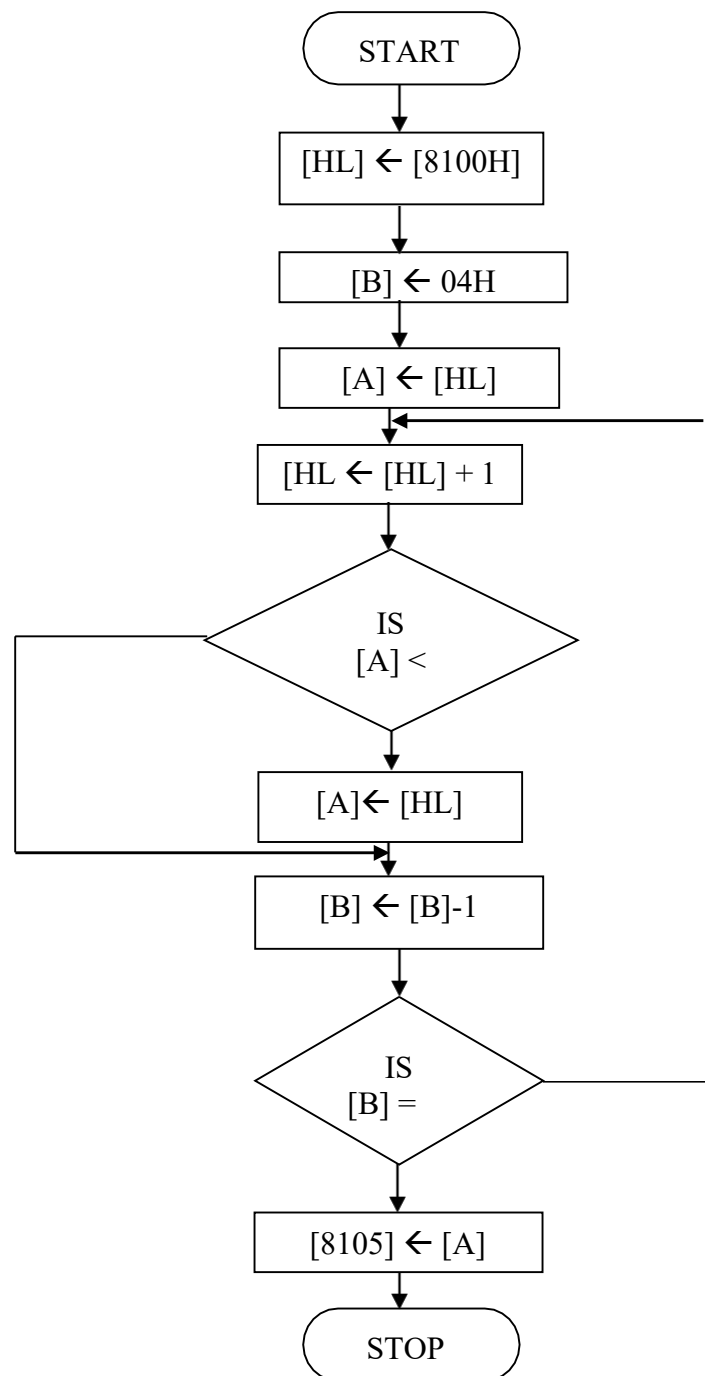
**ALGORITHM:**

1. Place all the elements of an array in the consecutive memory locations.
2. Fetch the first element from the memory location and load it in the accumulator.
3. Initialize a counter (register) with the total number of elements in an array.
4. Decrement the counter by 1.
5. Increment the memory pointer to point to the next element.
6. Compare the accumulator content with the memory content (next element).
7. If the accumulator content is smaller, then move the memory content (largest element) to the accumulator. Else continue.
8. Decrement the counter by 1.
9. Repeat steps 5 to 8 until the counter reaches zero
10. Store the result (accumulator content) in the specified memory location.

**RESULT:**

Thus the largest number in the given array is found out.

## FLOW CHART:



**PROGRAM:**

ADDRESS	OPERAND	LABEL	MNEMONICS	OPERAND	COMMENTS
8001			LXI	H,8100	Initialize HL reg. to 8100H
8002					
8003					
8004			MVI	B,04	Initialize B reg with no. of comparisons(n-1)
8005					
8006			MOV	A,M	Transfer first data to acc.
8007		LOOP1	INX	H	Increment HL reg. to point next memory location
8008			CMP	M	Compare M & A
8009			JNC	LOOP	If A is greater than M then go to loop
800A					
800B					
800C			MOV	A,M	Transfer data from M to A reg
800D		LOOP	DCR	B	Decrement B reg
800E			JNZ	LOOP1	If B is not Zero go to loop1
800F					
8010					
8011			STA	8105	Store the result in a memory location.
8012					
8013					
8014			HLT		Stop the program

**OBSERVATION:**

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
8100		8105	
8101			
8102			
8103			
8104			