Vision of the Department

To be recognized for keeping innovation, research and excellence abreast of learning in the field of computer science & engineering to cater the global society.

Mission of the Department

M1: To provide an exceptional learning environment with academic excellence in the field of computer science and engineering.

M2: To facilitate the students for research and innovation in the field of software, hardware and computer applications and nurturing to cater the global society.

M3: To establish professional relationships with industrial and research organisations to enable the students to be updated of the recent technological advancements.

M4: To groom the learners for being the software professionals catering the needs of modern society with ethics, moral values and full of patriotism.

Program Educational Objectives (PEO's)

PEO1: The graduate will have the knowledge and skills of major domains of computer science and engineering in providing solution to real world problems most efficiently.

PEO2: The graduate will be able to create and use the modern tools and procedures followed in the software industry in the relevant domain.

PEO3: The graduate will be following the ethical practices of the software industry and contributing to the society as a responsible citizen.

PEO4: The graduate will have the innovative mindset of learning and implementing the latest developments and research outcomes in the computer hardware and software to keep pace with the fast changing socio economic world.

Course OUTCOMES

CO1: Analyze instruction execution cycle and addressing modes for computer processor..

CO2: Analyze computer arithmetic and types of micro-processor.

CO3: Describe I/O subsystems

CO4: Draw Memory architecture with diagram

CO5: Interpret the use of parallel processing in uniprocessor system & multiprocessor architecture.

LIST OF EXPERIMENTS

1. Study of Multiplexer and Demultiplexer
2. . Study of Half Adder and Subtractor.
3. Study of Full Adder and Subtractor.
4. WAP to add two 8 bit numbers and store the result at memory location 2000
5. WAP to multiply two 8 bit numbers stored at memory location 2000 and 2001 and stores the result at memory location 2000 and 2001.
6. WAP to add two 16-bit numbers. Store the result at memory address starting from 2000.
7. WAP which tests if any bit is '0' in a data byte specified at an address 2000. If it is so, 00 would be stored at address 2001 and if not so then FF should be store data the same address.
8. Assume that 3 bytes of data are stored at consecutive memory addresses of the data memory starting at 2000. Write a program which loads register C with (2000), i.e. with data contained at memory address2000, D with (2001), E with (2002) and A with (2001).
9. Sixteen bytes of data are specified at consecutive data-memory locations starting at 2000. Write a program which increments the value of all sixteen bytes by 01.
10. WAP to add t 10 bytes stored at memory location starting from 3000. Store the result at memory location 30

Experiment: - 1

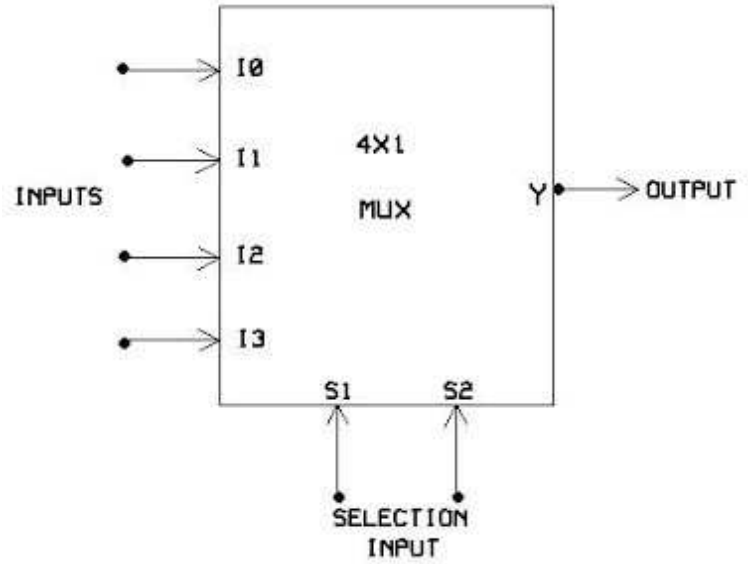
Aim: Study of Multiplexer and Demultiplexer.

Multiplexer is a digital switch which allows digital information from several sources to be routed onto a single output line. The basic multiplexer has several data input lines and a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2n input lines and n selector lines whose bit combinations determine which input is selected. Therefore, multiplexer is 'many into one' and it provides the digital equivalent of an analog selector switch.
A Demultiplexer is a circuit that receives information on a single line and transmits this information on one of 2n possible output lines. The selection of specific output line is controlled by the values of n selection lines.

DESIGN:

4 X 1 MULTIPLEXER

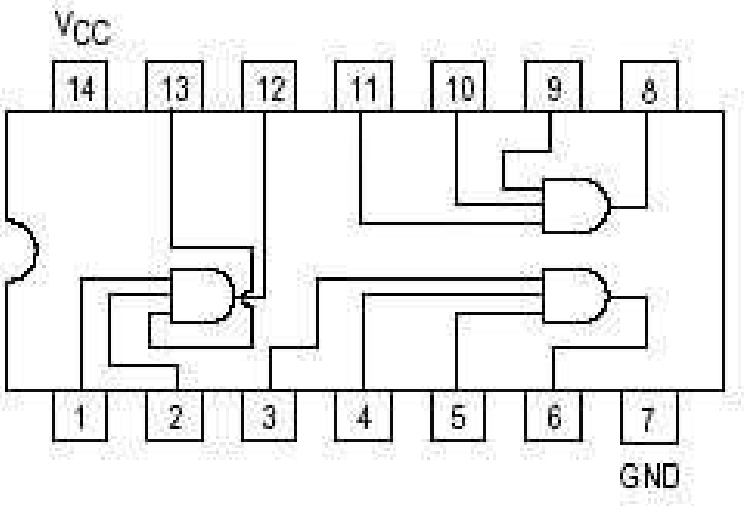
LOGIC SYMBOL:



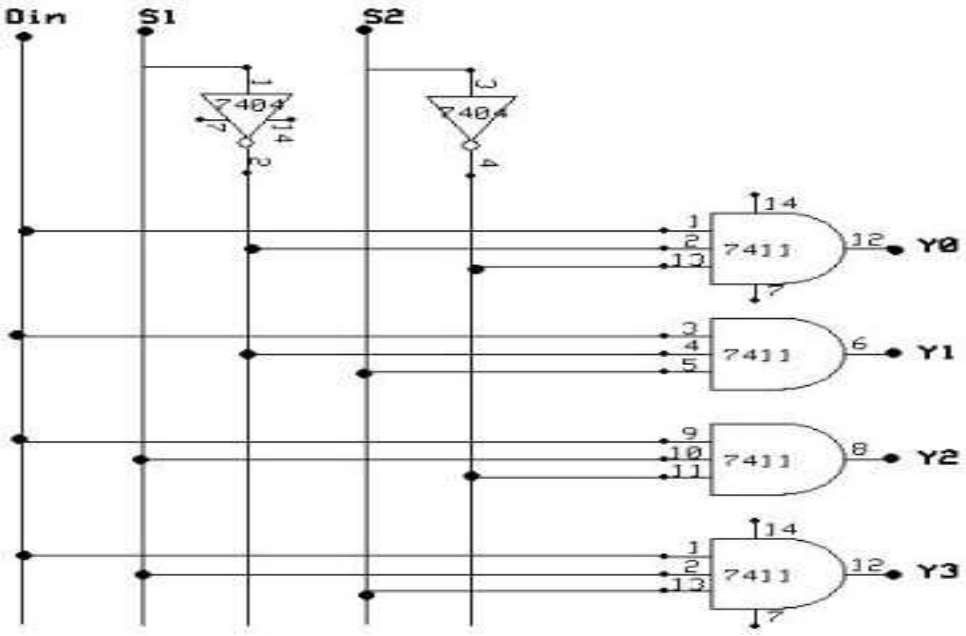
TRUTH TABLE:

S. No	SELECTION INPUT		OUTPUT
	S1	S2	Y
1.	0	0	I_0
2.	0	1	I_1
3.	1	0	I_2
4.	1	1	I_3

PIN DIAGRAM OF IC 7411:

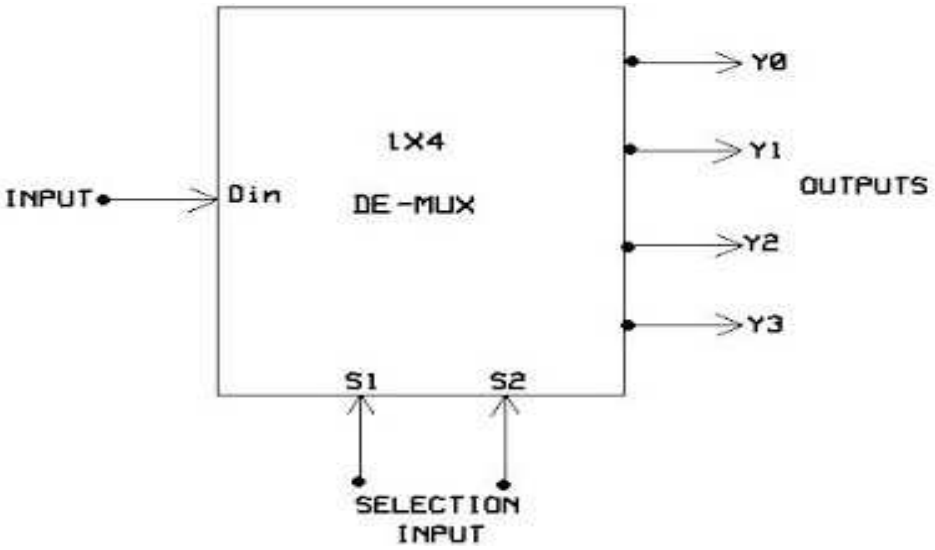


CIRCUIT DIAGRAM:



1X4 DEMULTIPLEXER

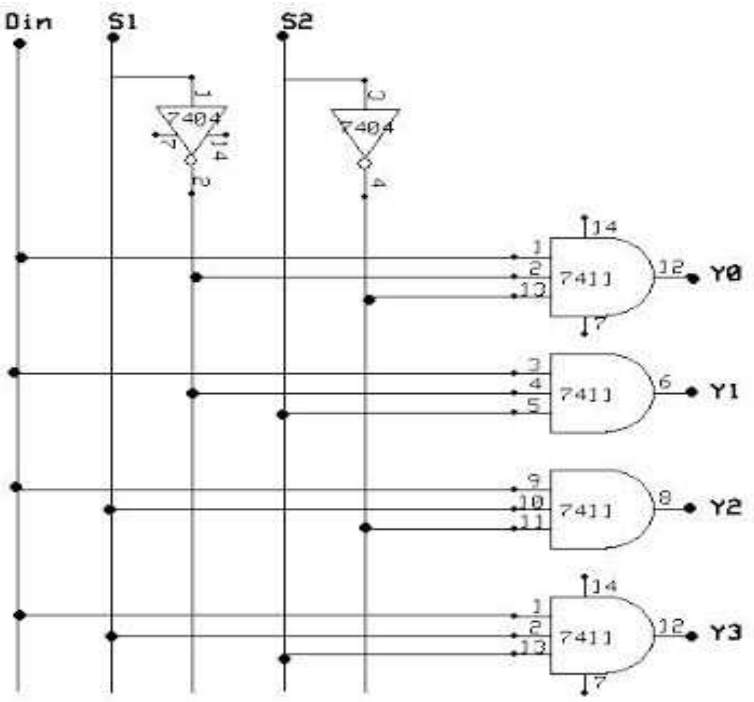
LOGIC SYMBOL:



TRUTH TABLE:

S. No	INPUT			OUTPUT			
	S1	S2	Din	Y0	Y1	Y2	Y3
1.	0	0	0	0	0	0	0
2.	0	0	1	1	0	0	0
3.	0	1	0	0	0	0	0
4.	0	1	1	0	1	0	0
5.	1	0	0	0	0	0	0
6.	1	0	1	0	0	1	0
7.	1	1	0	0	0	0	0
8.	1	1	1	0	0	0	1

CIRCUIT DIAGRAM



PROCEDURE:

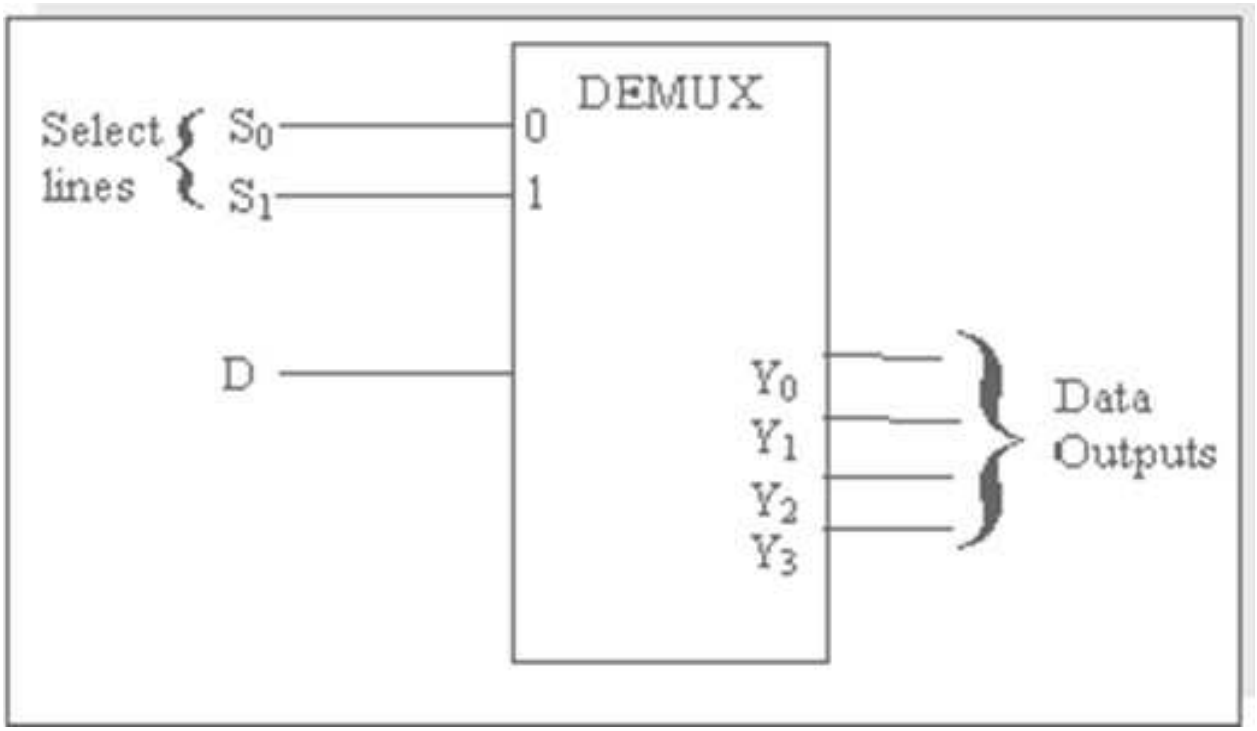
1. Connections are given as per the circuit diagrams.
2. For all the ICs 7^{th} pin is grounded and 14^{th} pin is given +5 V supply.
3. Apply the inputs and verify the truth table for the multiplexer & demultiplexer.

DEMULTIPLEXERS:

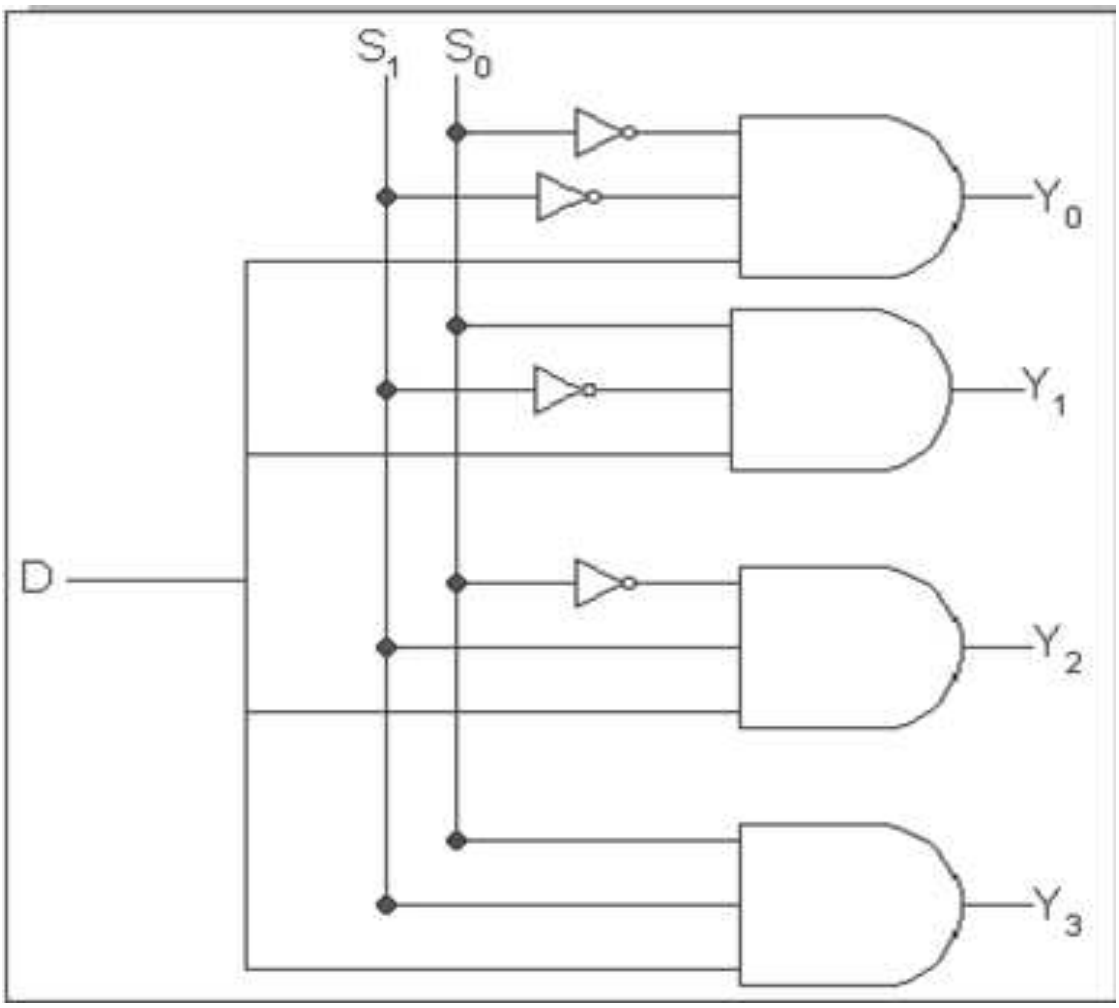
The Demultiplexer is combinational logic circuit that performs the reverse operation of Multiplexer. It has only one input, n selectors and 2n outputs. Depending on the combination of the select lines, one of the outputs will be selected to take the state of the input.

The following figure shows the block diagram and the truth table for 1x4 Demultiplexer.

By applying logic '1' to the input, the circuit will do the same function of the typical 2-to-4 Decoder.



Data	Address		Outputs			
	S_1	S_0	Y_0	Y_1	Y_2	Y_3
D	0	0	D	0	0	0
D	0	1	0	D	0	0
D	1	0	0	0	D	0
D	1	1	0	0	0	D


S1
S0
D
Y0
Y1
Y2
Y3

Experiment: - 2

Aim: Study of Half Adder and Subtractor.

Apparatus required:

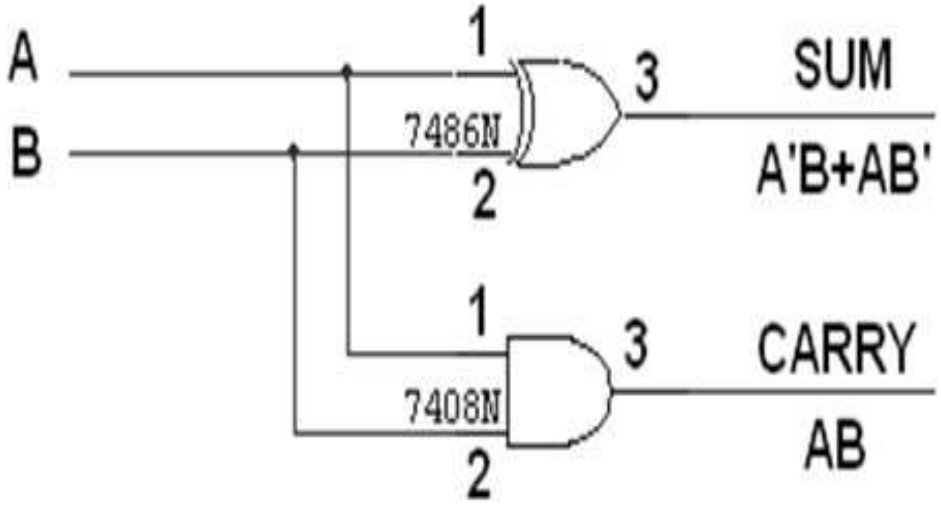
S. No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	1
2.	X-OR GATE	IC 7486	1
3.	NOT GATE	IC 7404	1
4.	OR GATE	IC 7432	1
5.	IC TRAINER KIT	-	1
6.	PATCH CORDS	-	23

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.

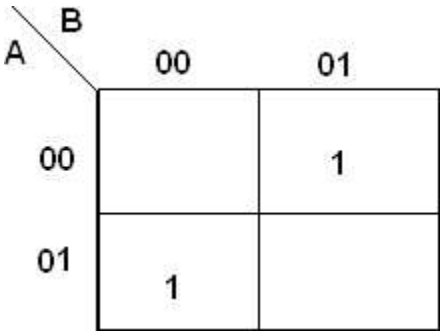
Logic: Half adder



TRUTH TABLE:

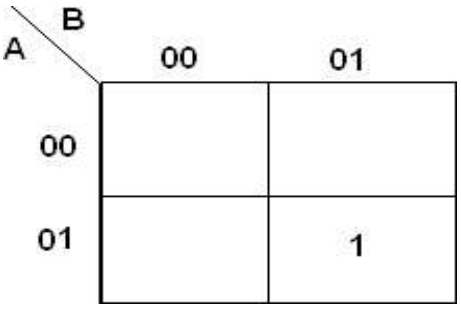
A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K-Map for SUM:



SUM = A'B + AB'

K-Map for CARRY:

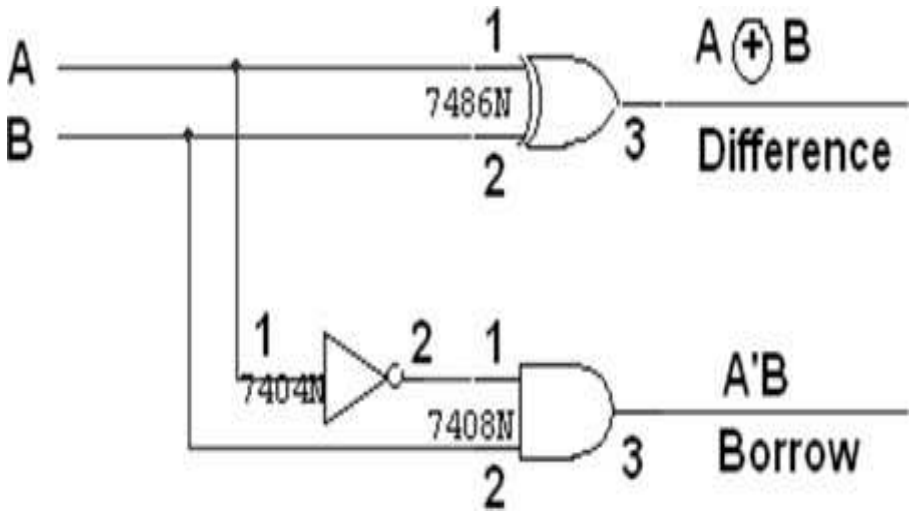


CARRY = AB

Logic diagram:

Half Subtractor:

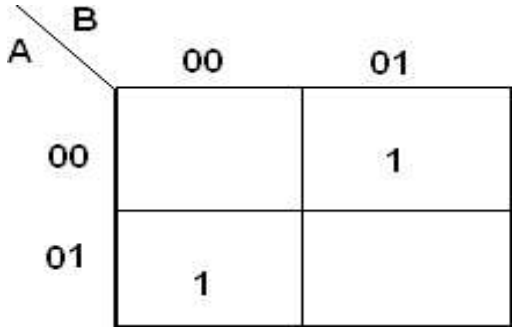
The half subtractor is constructed using X-OR and AND Gate. The half subtractor has two input and two outputs. The outputs are difference and borrow. The difference can be applied using X-OR Gate, borrow output can be implemented using an AND Gate and an inverter.



TRUTH TABLE:

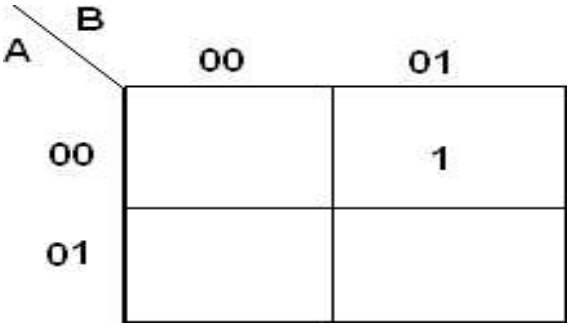
A	B	BORROW	DIFFERENCE
0	0	0	0
1	1	1	1
1	0	0	1
	1	0	0

K-map for difference:



DIFFERENCE = A'B + AB'

K-map for borrow:



BORROW = A'B

Experiment: - 3

Aim: Study of Full Adder and Subtractor.

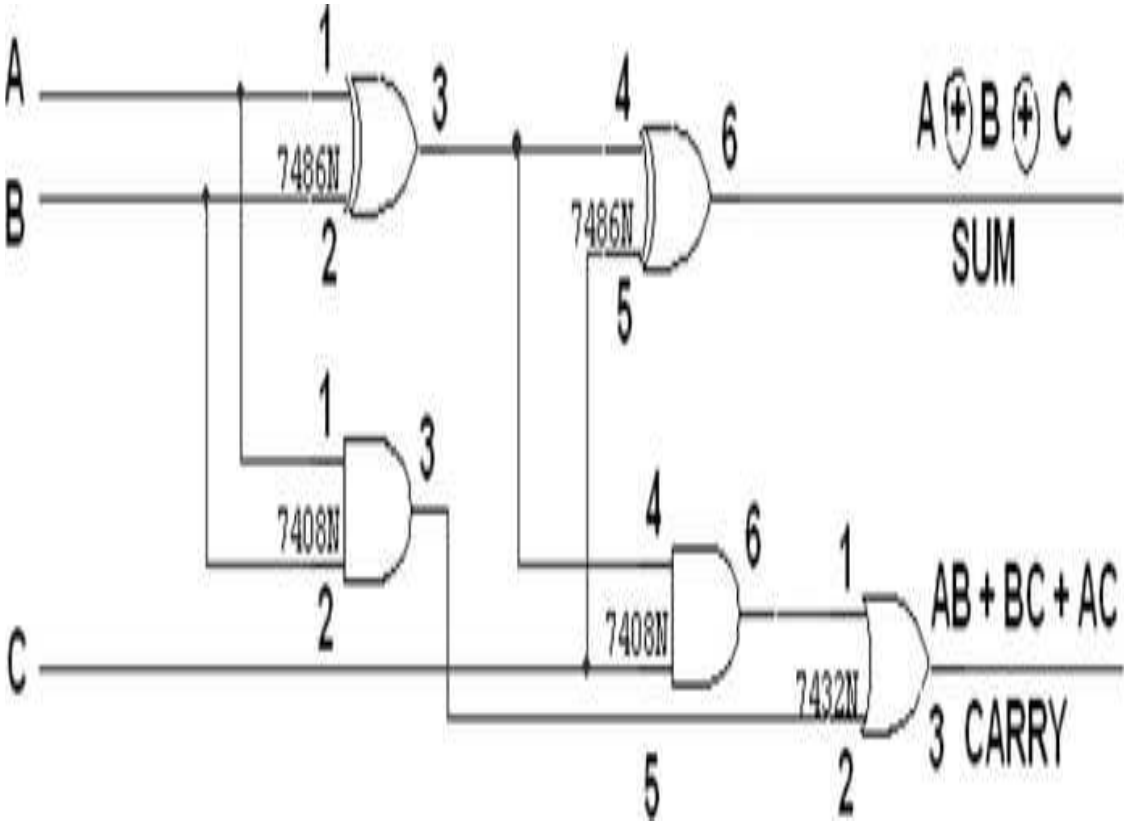
Full adder:

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:

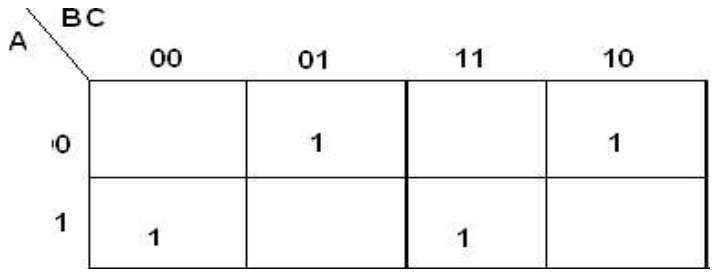
Full adder:

Full adder using half adder



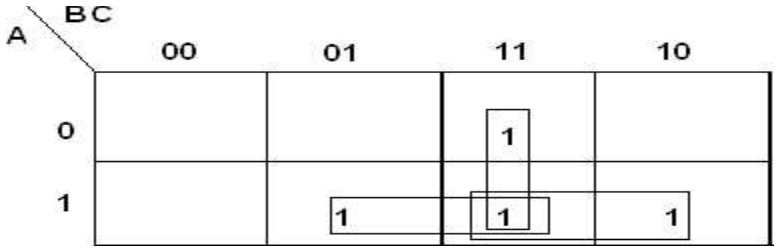
TRUTH TABLE:

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:

K-Map for CARRY



SUM = A'B'C + A'BC' + ABC' + ABC

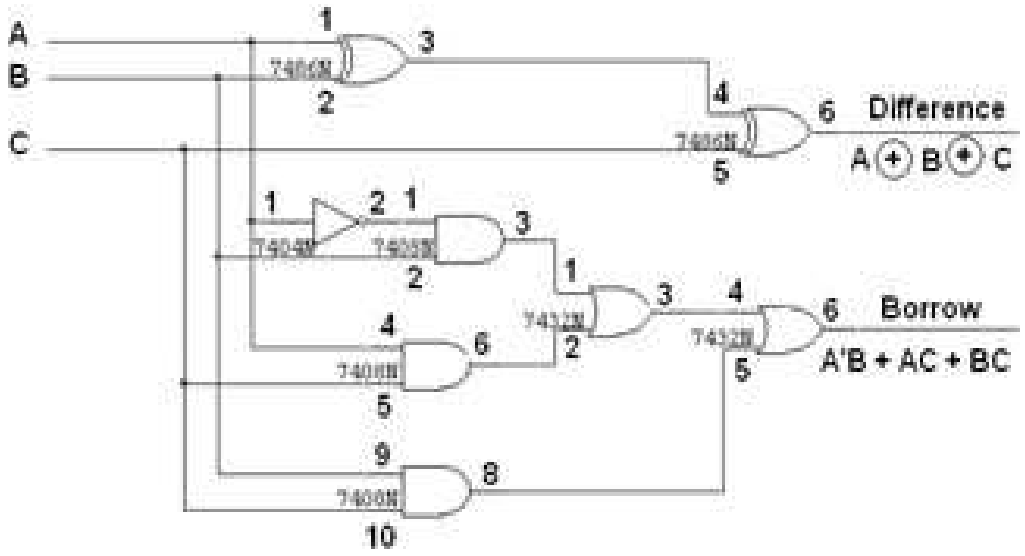
CARRY = AB + BC + AC

FULL SUBTRACTOR:

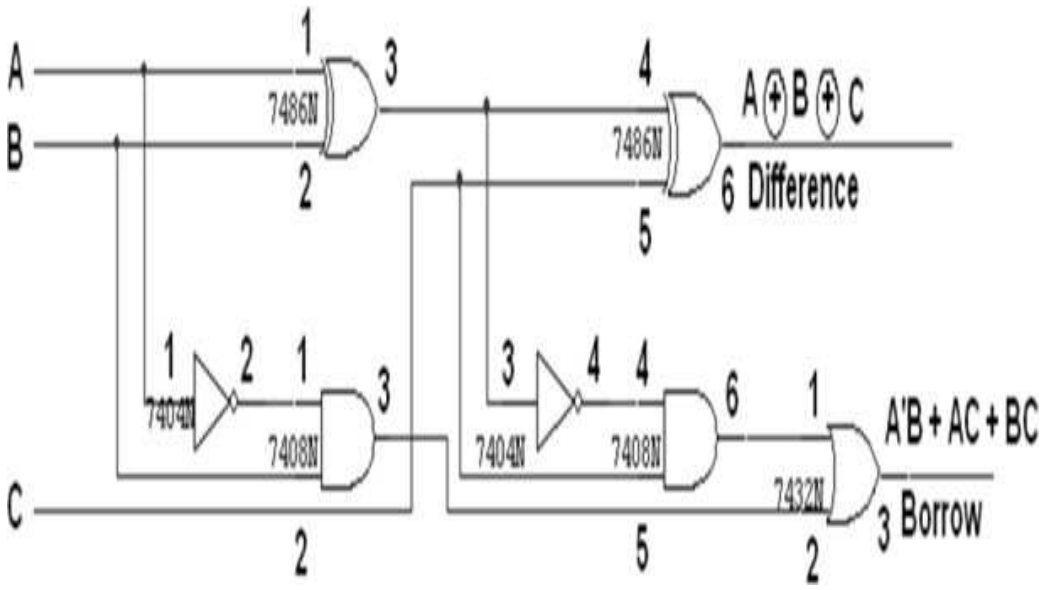
The full subtractor is a combination of X-OR, AND, OR, NOT Gates. In a full subtractor the logic circuit
should have three inputs and two outputs. The two half subtractor put together gives a full subtractor .The first half subtractor will be C and A B. The output will be difference output of full subtractor.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

FULL SUBTRACTOR



FULL SUBTRACTOR USING TWO HALF SUBTRACTOR:



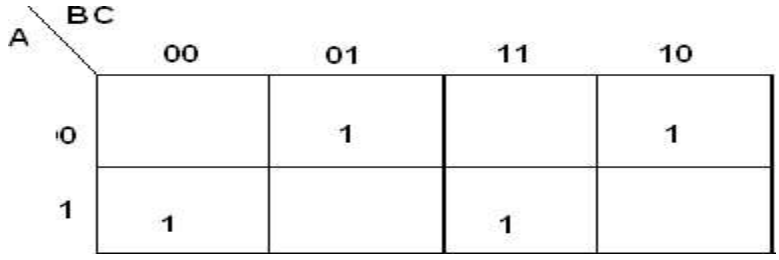
TRUTH TABLE:

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

C.O.A

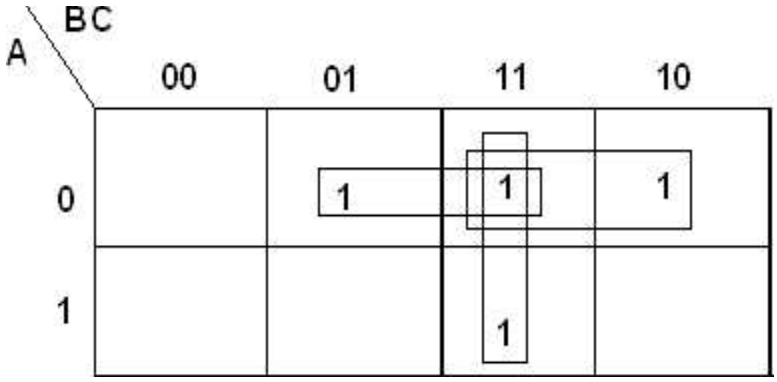
K-Map for Difference:

Difference = A’B’C + A’BC’ + AB’C’ + ABC



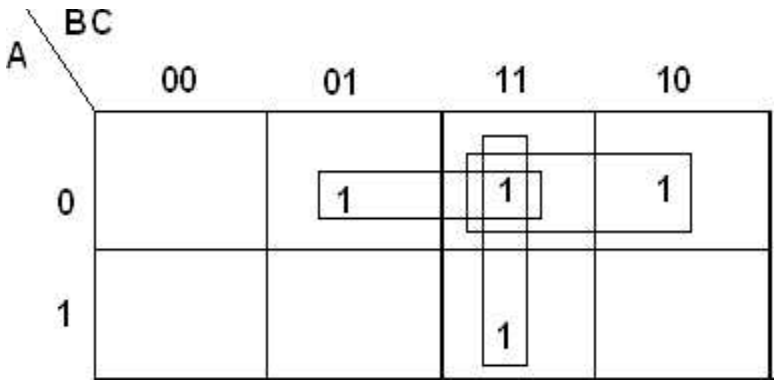
K-Map for Borrow

Borrow = A’B + BC + A’C



PROCEEDURE:

1. Connections are given as per circuit diagram.
2. Logical inputs are given as per circuit diagram.
3. Observe the output and verify the truth table.



CS-404

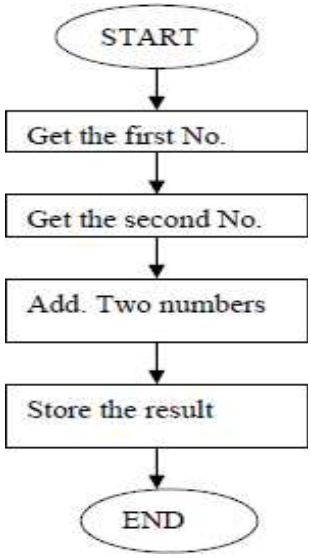
C.O.A

Experiment: - 4

Aim: WAP to add two 8 bit numbers and store the result at memory location 2000.

Apparatus required:

Microprocessor-8085 Trainer kit



Assume Two 8 Bit No Are; 05H & 23H	
Program	
MVI A, 05H	;Move 05h in to register A
MVI B, 23H	;Move 23h into register B
ADD A,B	; A ⇓---A + B
LXI H,2000H	;Load register pair HL with 2000H
MOV M,A	;Store the result of an Accumulator to the memory Whose address specified by HL register
HLT	
RESULT	Check the status of Register which are used in Program B=23H A=28H

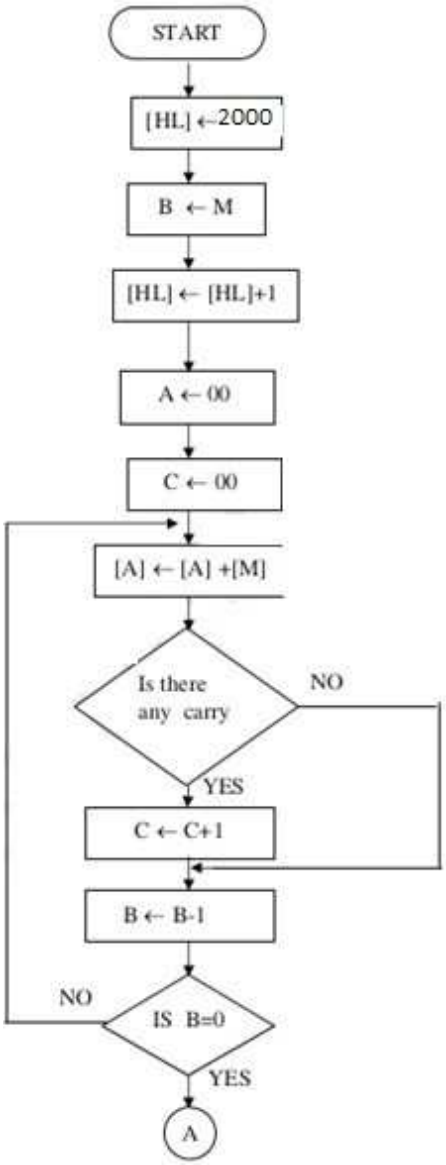
CS-404
C.O.A

Experiments- 5

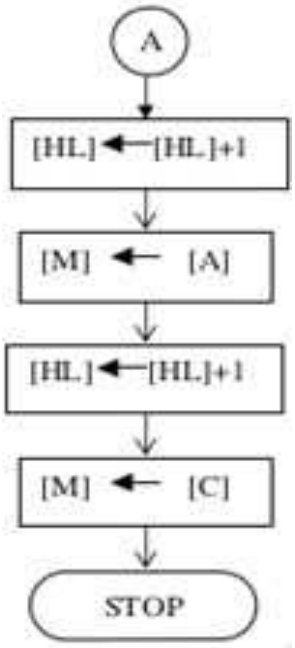
Aim: WAP to multiply two 8 bit numbers stored at memory location 2000 and 2001 and stores the result at memory location 2000 and 2001.

Apparatus Required:

Microprocessor-8085 Trainer kit



CS-404
C.O.A



	LXI H,2000H	
	MOV,B,M	
	INX H	
	MVI A,00H	
	MVI C,00H	
L1:	ADD M	
	JNC NXT	
	INR C	
NXT:	DCR B	
	JNZ L1	
	INX H	
	MOV M,A	
	INX H	
	MOV M,C	
	HLT	
RESULT	Check the status of the memory i. Before execu- tion of the pro- gram ii. After execu- tion of the pro- gram	

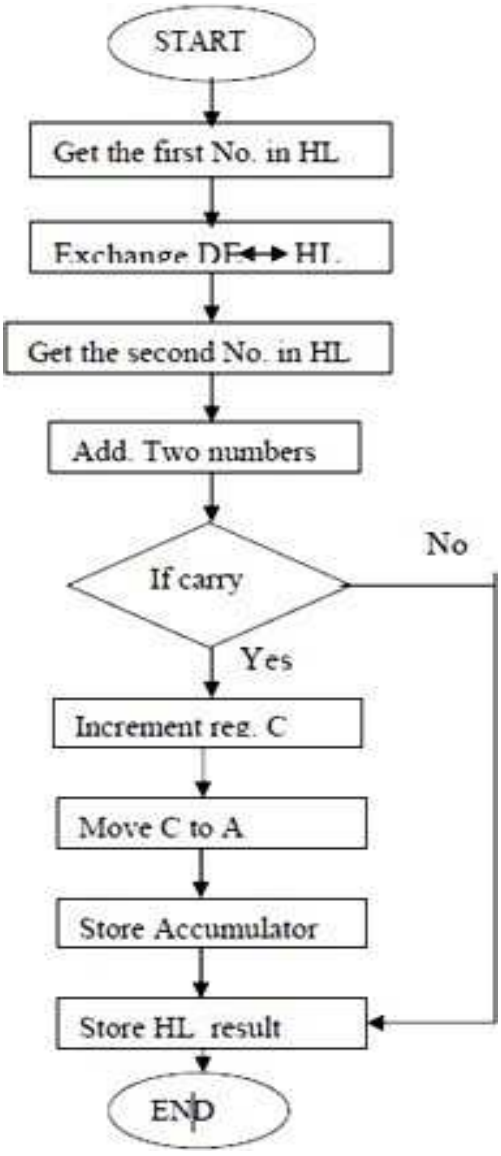
CS-404
C.O.A

Experiment: - 6

Aim: WAP to add two 16-bit numbers. Store the result at memory address starting from 2000.

Apparatus Required:

Microprocessor-8085 Trainer kit



LHLD	2000H
XCHG	
LHLD	2002
MVI	C,00H
DAD	D
JNC	LOOP
INR	C
MOV	A,C
STA	2000
LOOP; SHLD	
HLT	
RESULT	1. Check the status of memory at memory location 2000H, 2001H, 2002H, 2003H before the execution. 2. Check the status of memory at memory location 2000H,2001H after the execution

CS-404
C.O.A

Experiment: - 7

Aim: WAP which tests if any bit is '0' in a data byte specified at an address 2000. If it is so, 00 would be stored at address 2001 and if not so then FF should be stored at the same address**.**

Apparatus Required:

Microprocessor-8085 Trainer kit

LDA 2000H	Load Accumulator
JNZ LABEL1	Check the status of Accumulator
MVI A,00H	
STA 2001H;	If Data is zero display 00H at memory 2001
LABEL1: MVI A,FFH;	
STA 2001H	If Data is Non-zero Display FFh at memory 2001
HLT	
RESULT	If Data is zero display 00H at memory 2001 If Data is Non-zero Display FFh at memory 2001

CS-404
C.O.A

Experiment: - 8

Aim: Assume that 3 bytes of data are stored at consecutive memory addresses of the data memory starting at 2000. Write a program which loads register C with (2000), i.e. with data contained at memory address2000, D with (2001), E with (2002) and A with (2001).

Apparatus Required:

Microprocessor-8085 Trainer kit

Algorithm

Start
Load HL pair with 2000

Copy data from M to accumulator

Increment address in HL pair Copy data

from M to reg D Increment address in

HL pair Copy data from M to reg E

Load accumulator with the data byte at 2001

HLT

PROGRAM CODE:

```
LXI H,  2000
MOV C,M

INX H
MOV D,M
INX H
MOV E,M
LDA 2001
HLT
```

CS-404
COA

Experiment: - 9

Aim: Sixteen bytes of data are specified at consecutive data-memory locations starting at 2000. Write a program which increments the value of all sixteen bytes by 01**.**

Appratus required:

Microprocessor-8085 Trainer kit

Algorithm:

Load the starting address of the data memory into the H and L registers.

Set the loop counter to 16, and store it in a register, e.g., register B.

Start a loop that iterates 16 times.

Load the value at the current memory location (pointed to by H and L) into the accumulator (register A).

Increment the value in the accumulator by 1.

Store the updated value back to the current memory location.

Move to the next memory location by incrementing the high byte of the memory address (register H).

Decrement the loop counter (register B).

Check if the loop counter is not zero.

If the loop counter is not zero, jump back to step 4 (looping).

If the loop counter is zero, halt the program.

PROGRAM CODE:

LXI H, 2000h;

MOV H, 20h;

MOV L ,00h

MVI B, 16;

LOOP_START:

MOV A, M;

INR A;

MOV M , A;

INX H;

DCR B;

JNZ LOOP_START;

HLT;

Experiment: - 10

Aim: WAP to add t 10 bytes stored at memory location starting from 3000. Store the result at memory location 300A.

ALGORITHM:

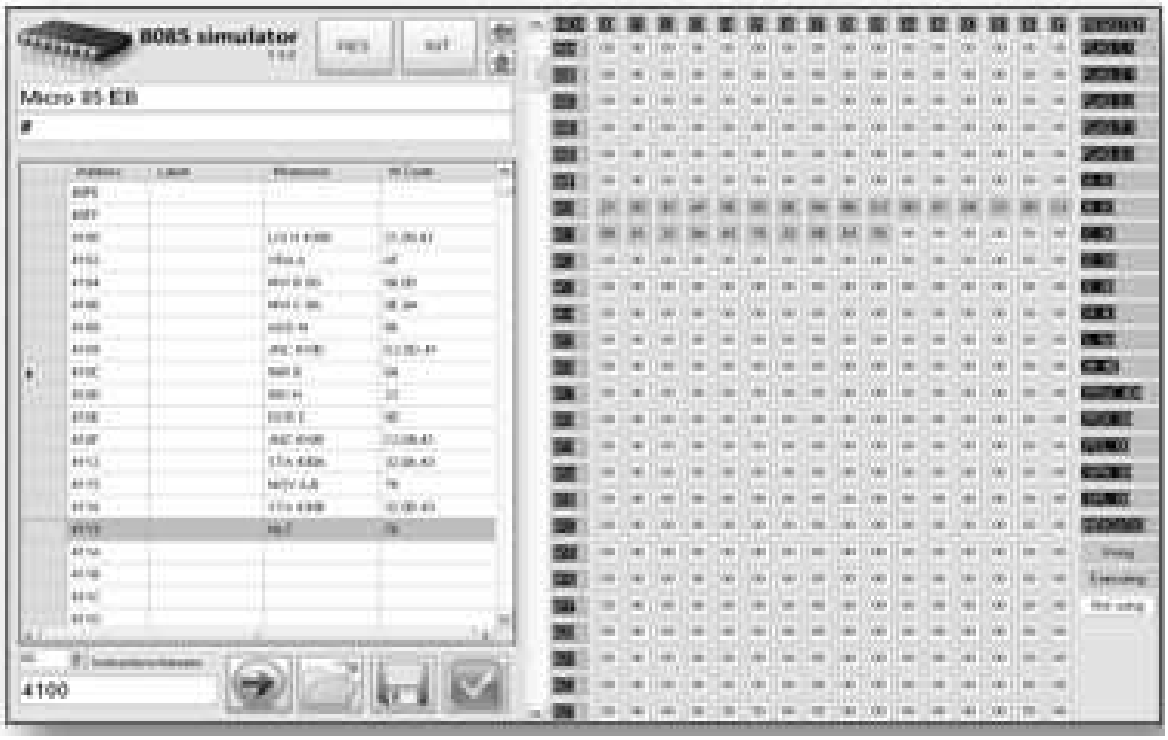
1. Load HL register with the first source byte address.
2. Clear Accumulator
3. Clear Register B
4. Load C with count $10_{10}=0A_{16}$
5. Add byte
6. Go to 7 if No Carry
7. Increment B as the ere is a carry
8. Fetch next byte by incrementing HL pair
9. Decrement Count by 1
10. Jump if not zero to step 5
11. Store the accumulator content (lower byte of sum) at 300A
12. Stop

PROGRAM CODE:

```
LXI H, 3000 XRA  A
MVI B, 00 MVI C,
0A
 NEXTBYTE  ADD
  JNC AHEAD
   INR B AHEAD
    INX HDCR C
    JNZ NEXT
      HLT
```

CS-404
COA

OUTPUT :



Following bytes were saved at memory location 4130 onwards:57, 68 ,36 ,F2,57,E6,57,8A,12,24,91,79,35,46,24,57.