**SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING**

**(AUTONOMOUS)**

**RAJIV GANDHI SALAI, KALAVAKKAM**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

|  |  |
| --- | --- |
| **Name** | S.Ashwath |
| **Register Number** | 195002015 |
| **Class & Section** | IT-A |
| **Subject Code** | UIT1411 |
| **Subject Name** | Microprocessor and Microcontroller Laboratory |

**Sign of Faculty in-charge**

**Name : S.Ashwath**

**Register Number : 195002015**

**Subject Code :** UIT1411

**Subject Name :** Microprocessor and Microcontroller Laboratory

**INDEX**

DIGITAL CIRCUITS

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO** | **DATE** | **EXPERIMENT NAME** | **PAGE NO** |
| 1 |  | JOHNSON COUNTER | 4 |
| 2 |  | UP / DOWN COUNTER | 7 |
| 3 |  | ENCODER AND DECODER | 13 |
| 4 |  | HALF AND FULL ADDER | 19 |
| 5 |  | SR AND JK FLIP FLOP | 27 |

8086 PROGRAMS

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  | ADDITION OF TWO 32 BIT NUMBERS | 36 |
| 2 |  | MULTIPLICATION AND DIVISION OF TWO 16-BIT NUMBERS | 39 |
| 3 |  | ADDING N DECIMAL NUMBERS USING DAA | 43 |
| 4 |  | MOVING A LIST OF NUMBERS FROM ONE LOCATION TO ANOTHER | 46 |
| 5 |  | FIND THE SMALLEST NUMBER IN THE GIVEN SET OF NUMBERS | 49 |
| 6 |  | CONVERSION OF HEXADECIMAL TO DECIMAL NUMBER | 52 |
| 7 |  | CONVERSION OF DECIMAL TO HEXADECIMAL NUMBER | 55 |
| 8 |  | SORTING A LIST OF NUMBERS | 58 |
| 9 |  | FINDING THE COUNT OF A PARTICULAR BYTE IN LIST OF BYTES | 61 |
| 10 |  | DISPLAYING A STRING | 64 |

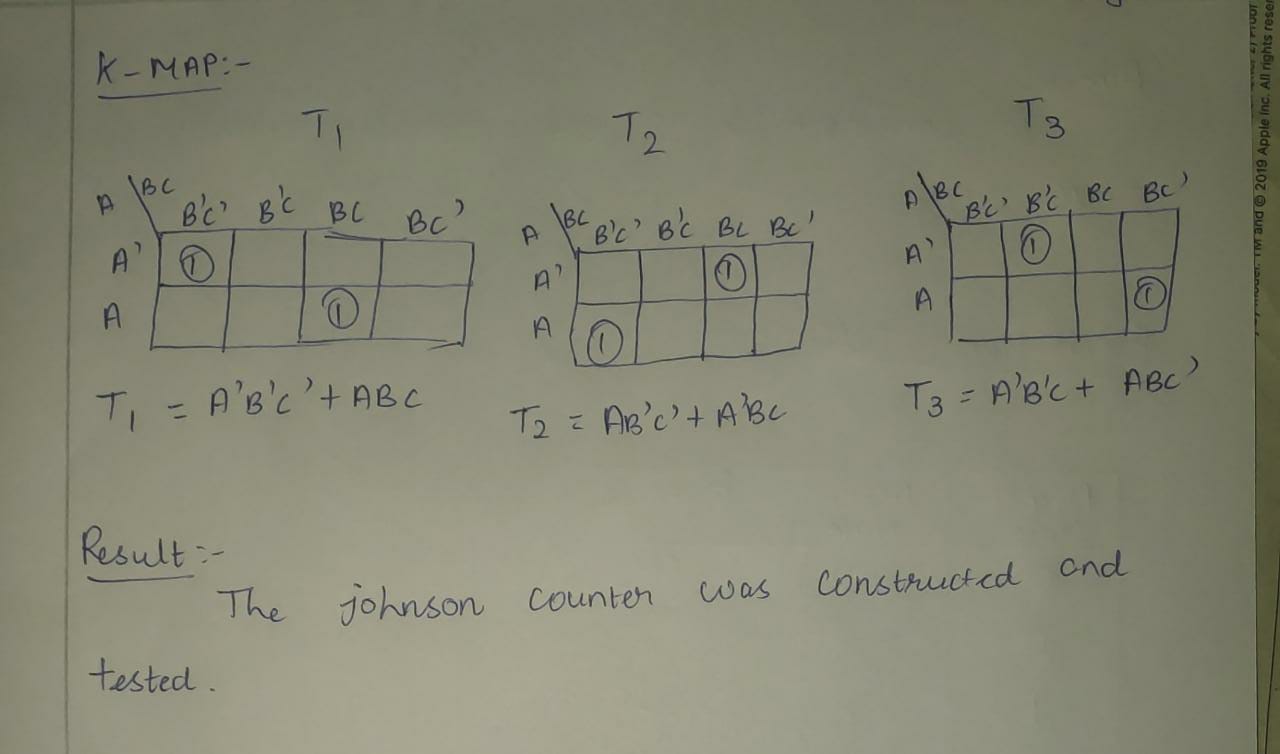
8051 PROGRAMS

|  |  |  |  |
| --- | --- | --- | --- |
| SNO | DATE | EXPERIMENT NAME | PAGE NO |
| 1 |  | COUNTING THE NUMBER OF OCCURANCE OF AN ELEMENT | 68 |
| 2 |  | COVERSION OF BINARY CODE TO GREY CODES | 70 |
| 3 |  | COPYING A BLOCK OF DATA FROM ONE LOCATION TO OTHER | 73 |
| 4 |  | COVERTING DECIMAL TO HEXADECIMAL NUMBER | 75 |
| 5 |  | CONVERTING HEXADECIMAL TO DECIMAL NUMBER | 77 |
| 6 |  | LOOKUP TABLE | 80 |
| 7 |  | FIND THE NUMBER OF ODD AND EVEN NUMBER IN THE LIST OF NUMBERS | 83 |
| 8 |  | BIT MANUPULATION INSTRUCTION | 86 |
| 9 |  | DECIMAL ADDTION AND CLEAR ACCUMULATOR AFTER 10MS | 88 |

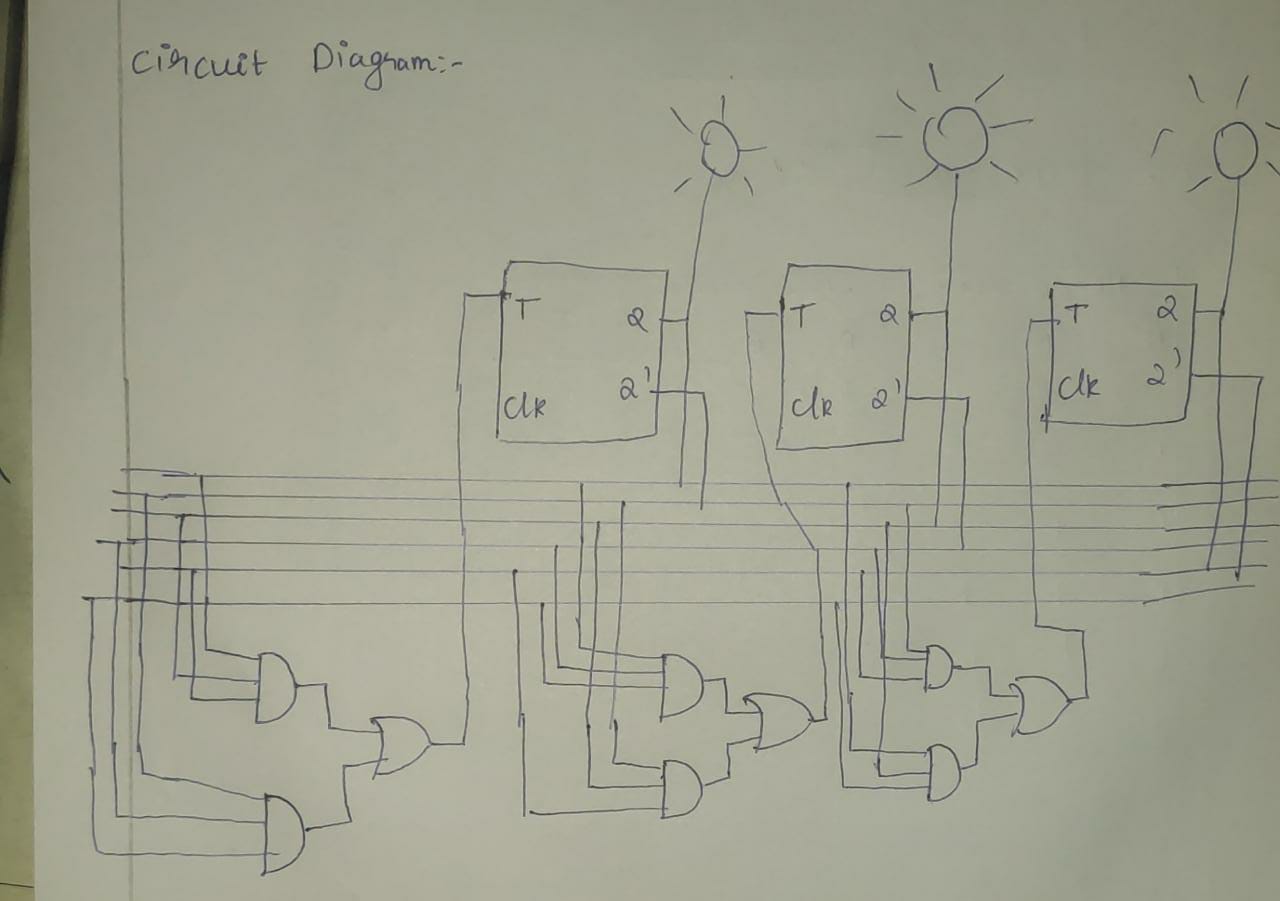
**Experiment Number: 1**

**Johnson counter**

****

****

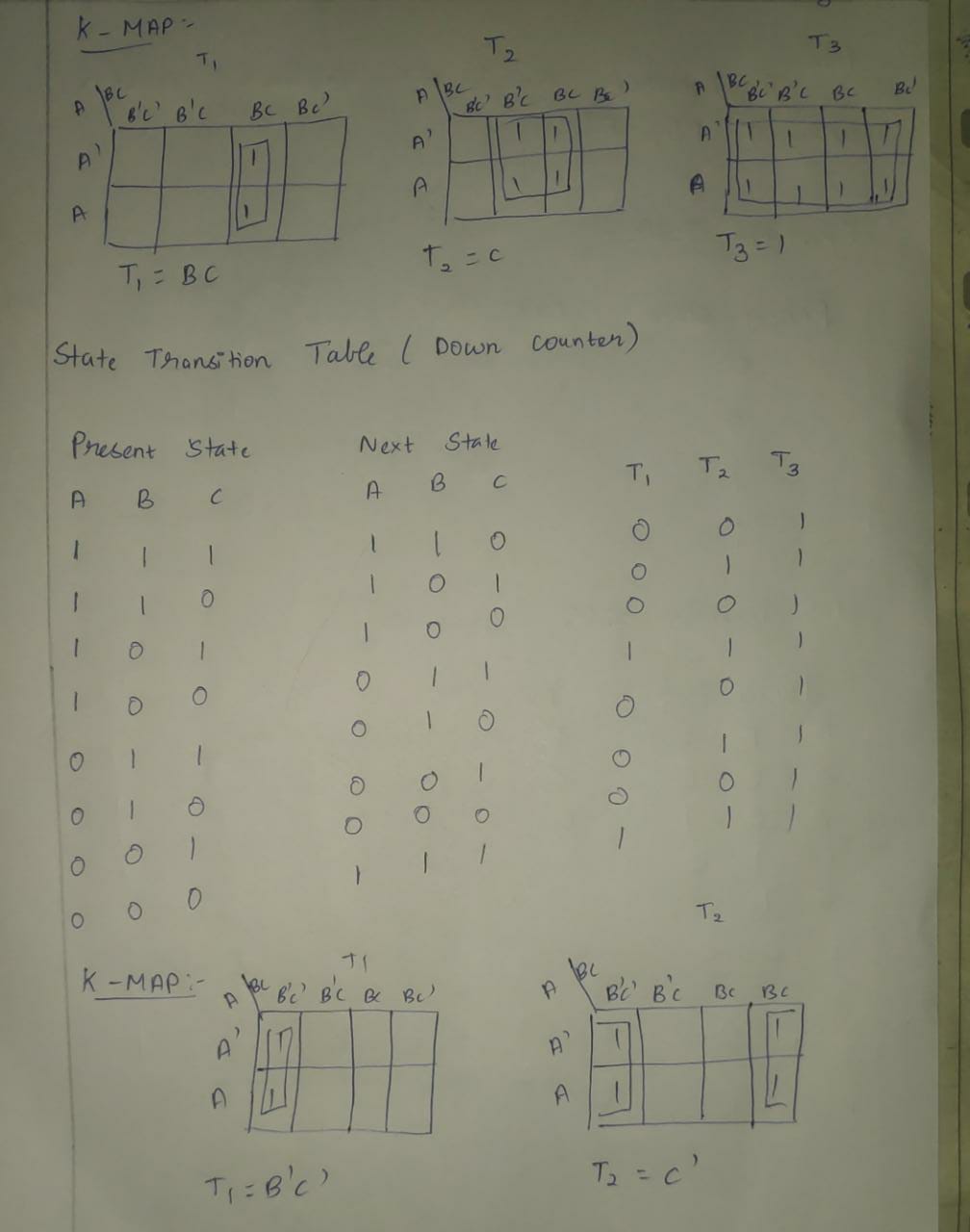
**Circuit Diagram:**

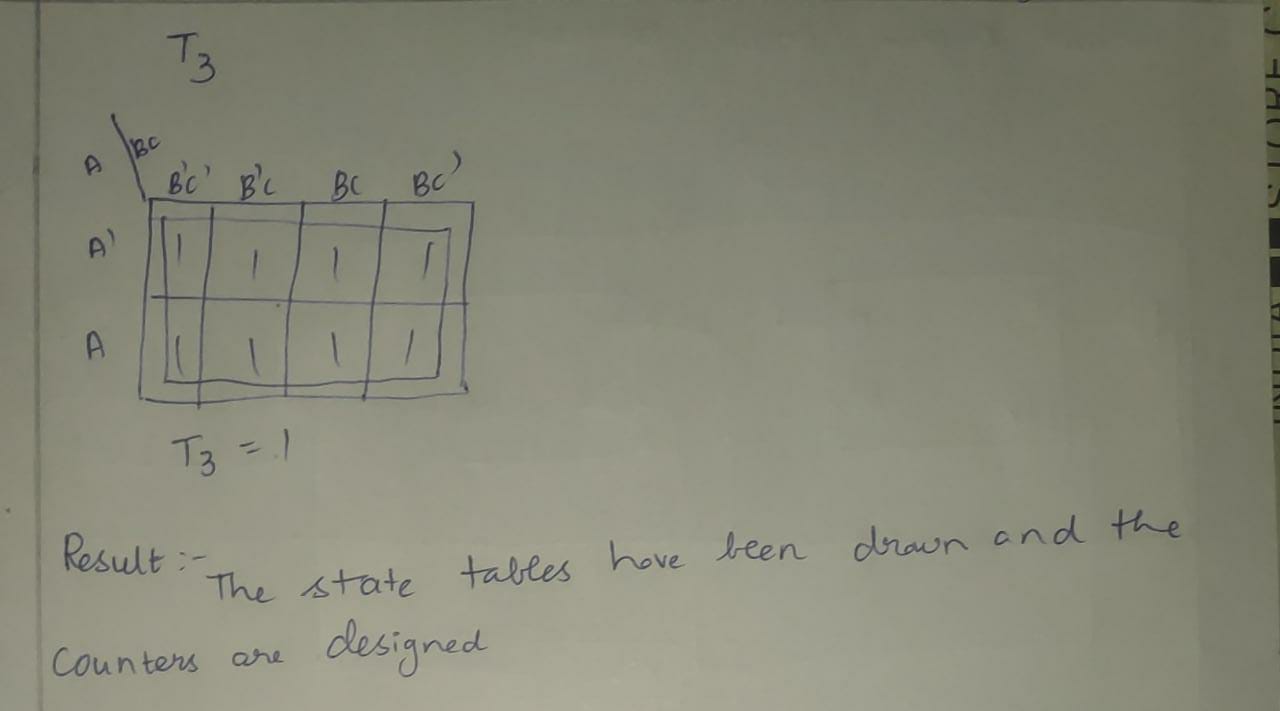


**Experiment Nnumber:2**

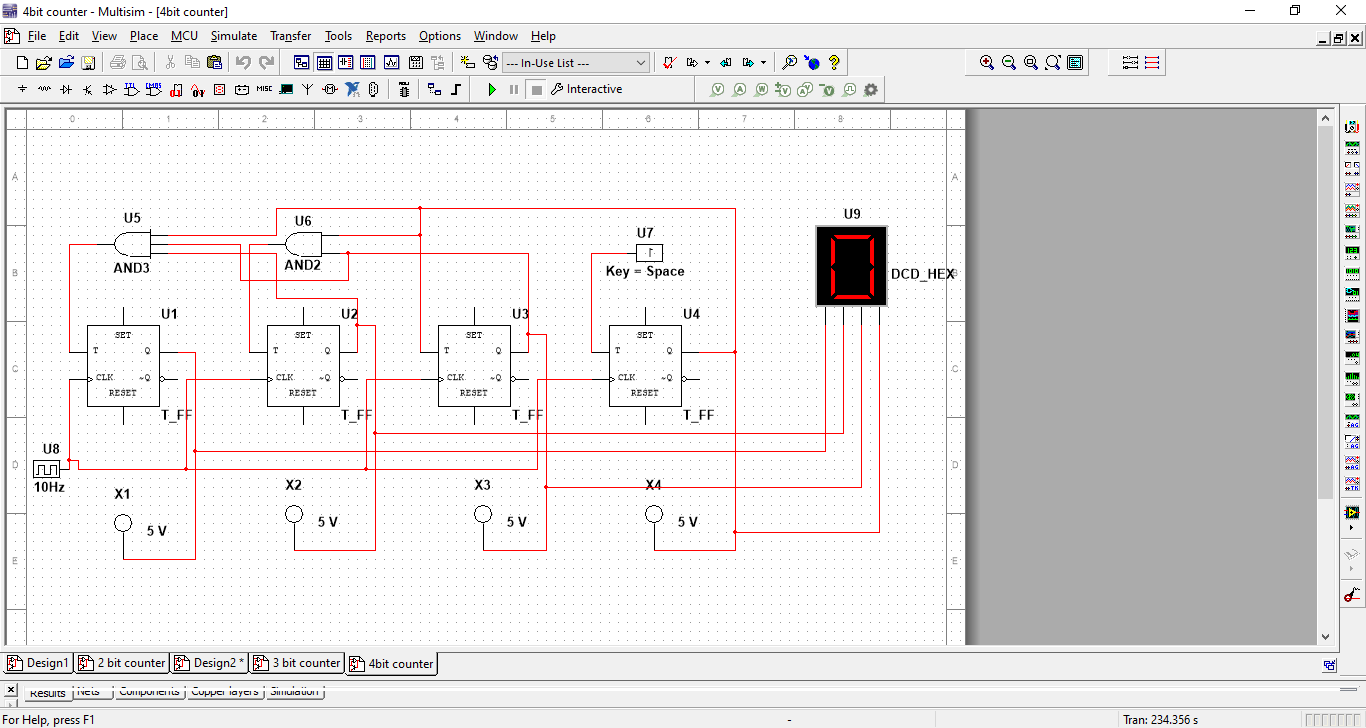
**Up/Down Counter**

****

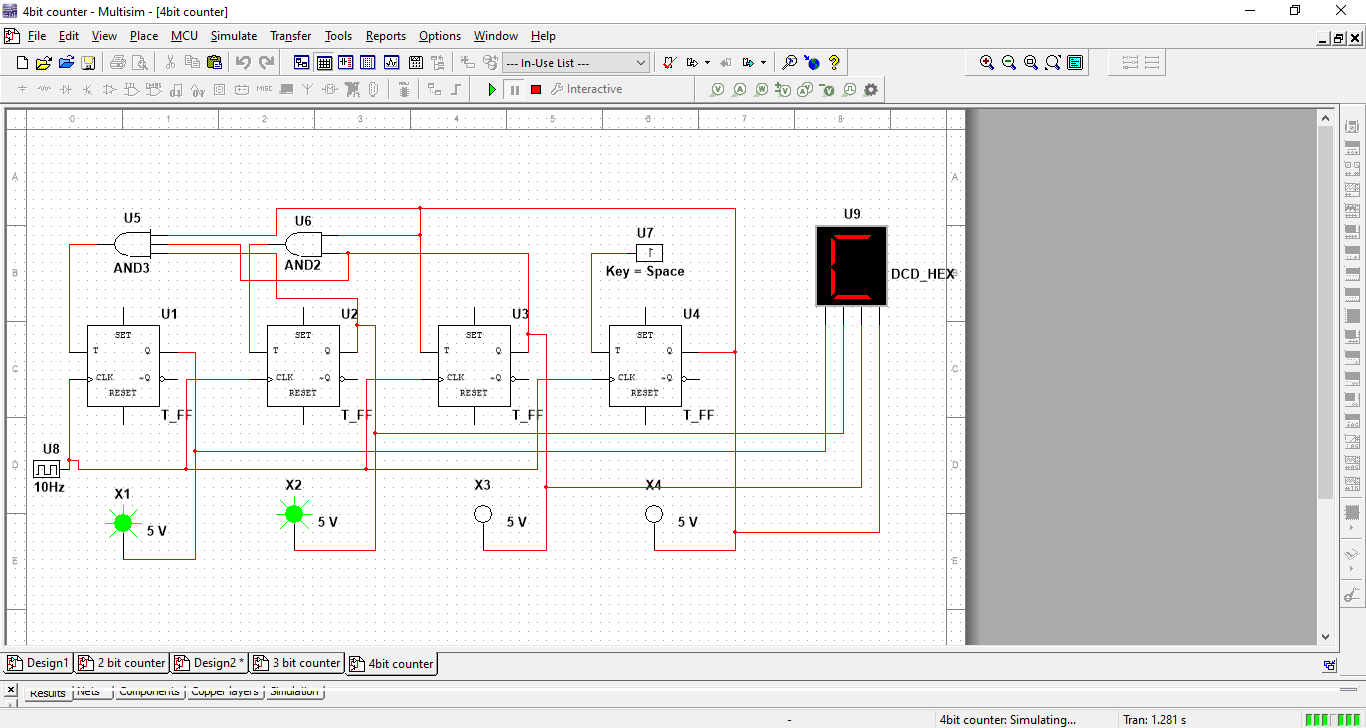
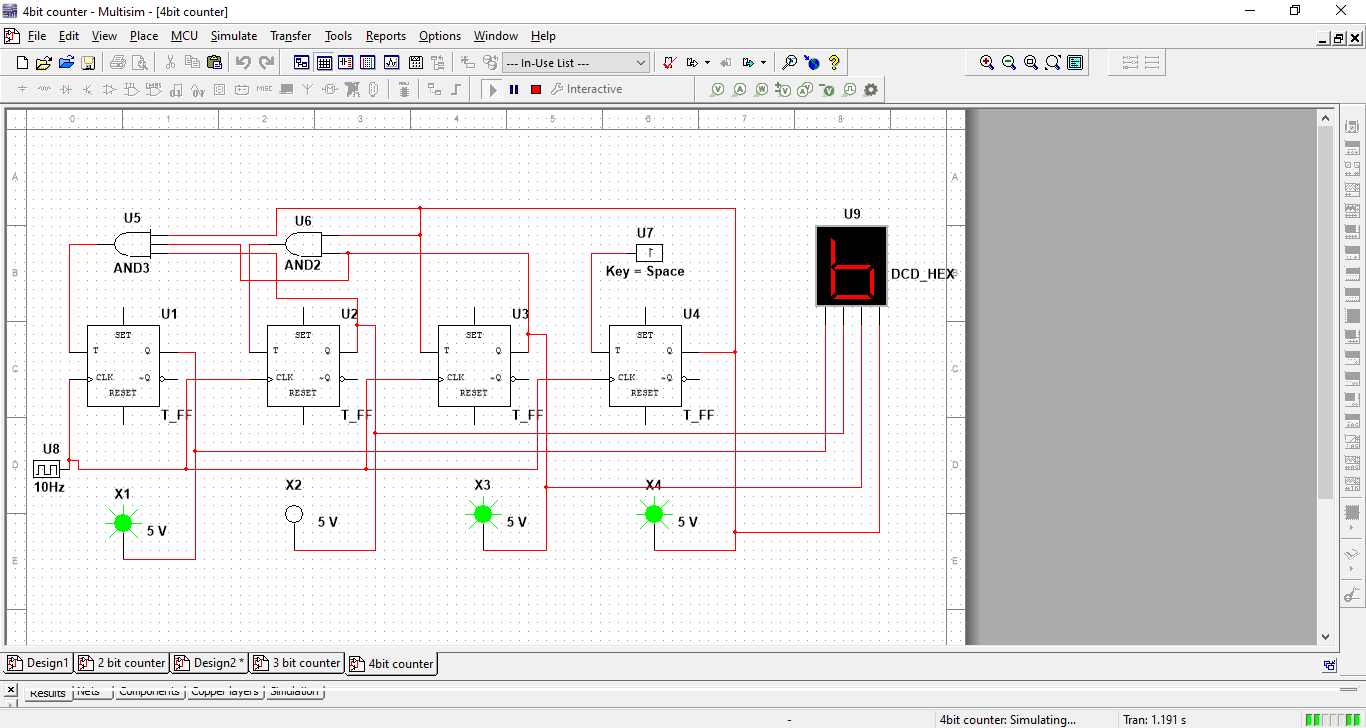
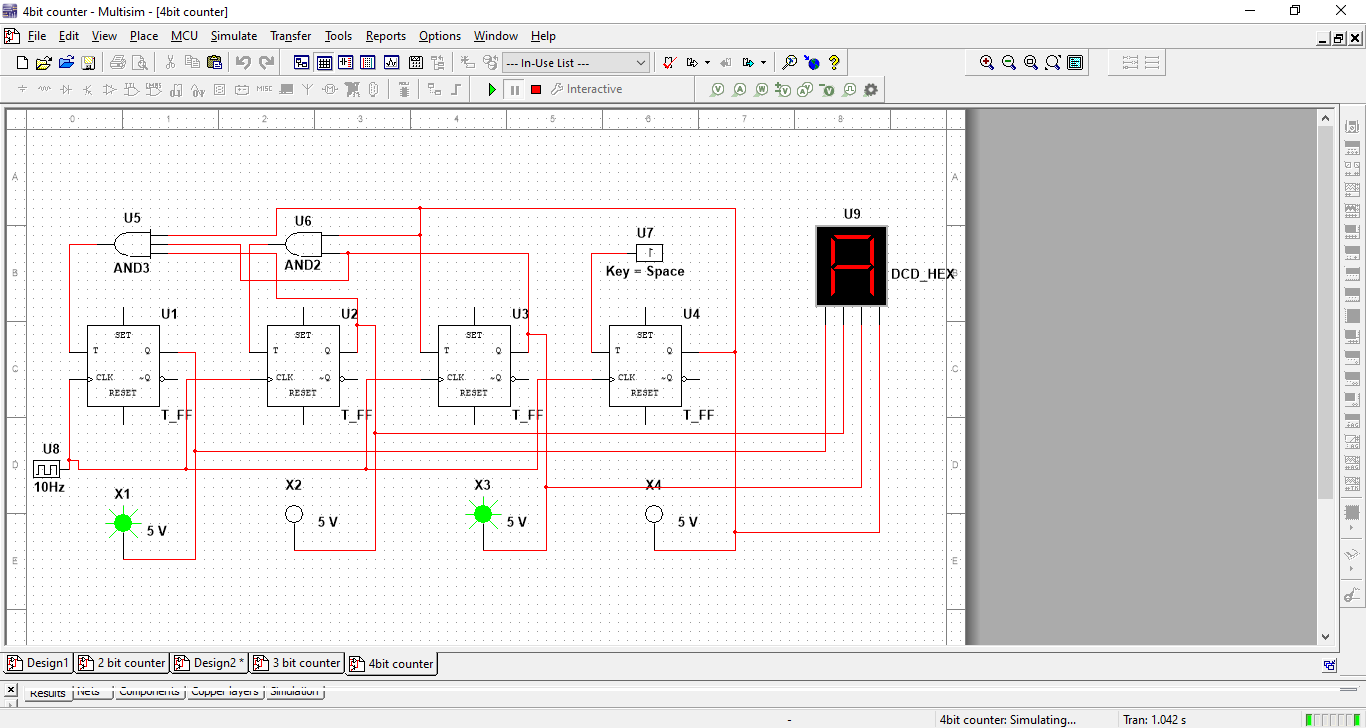
****

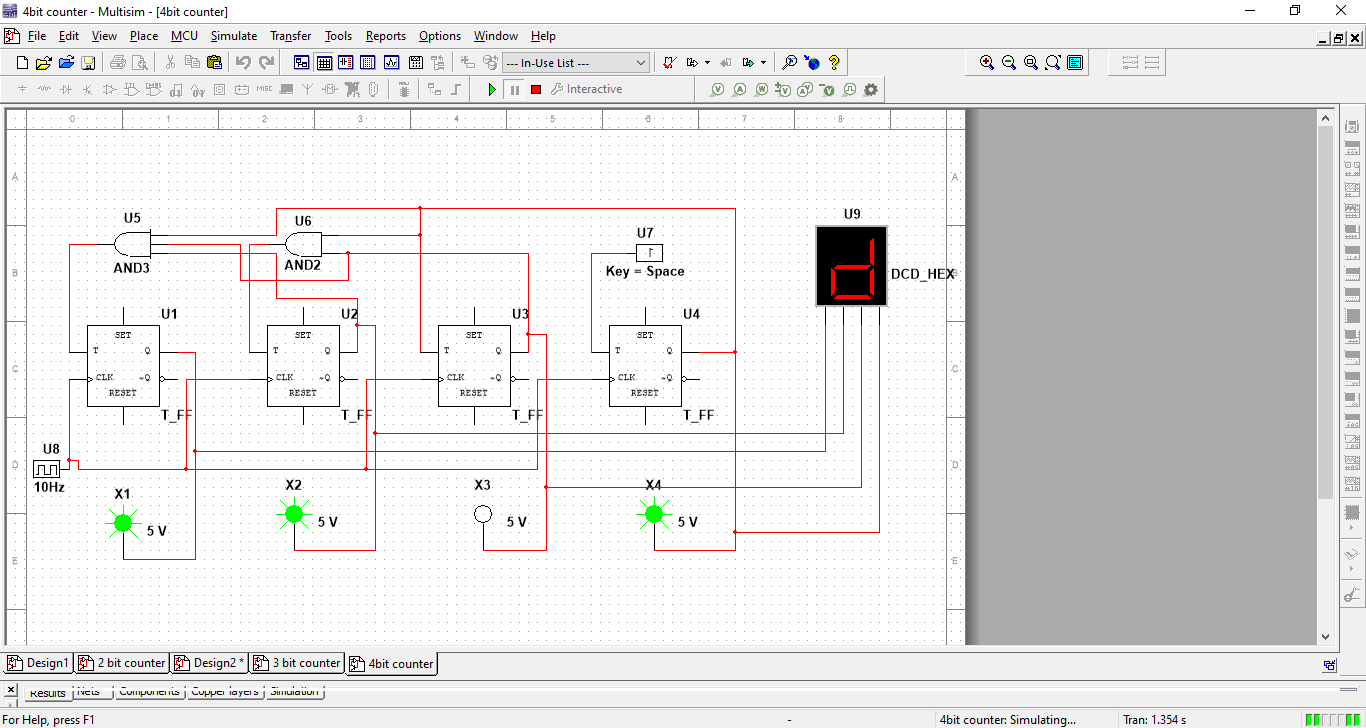
****

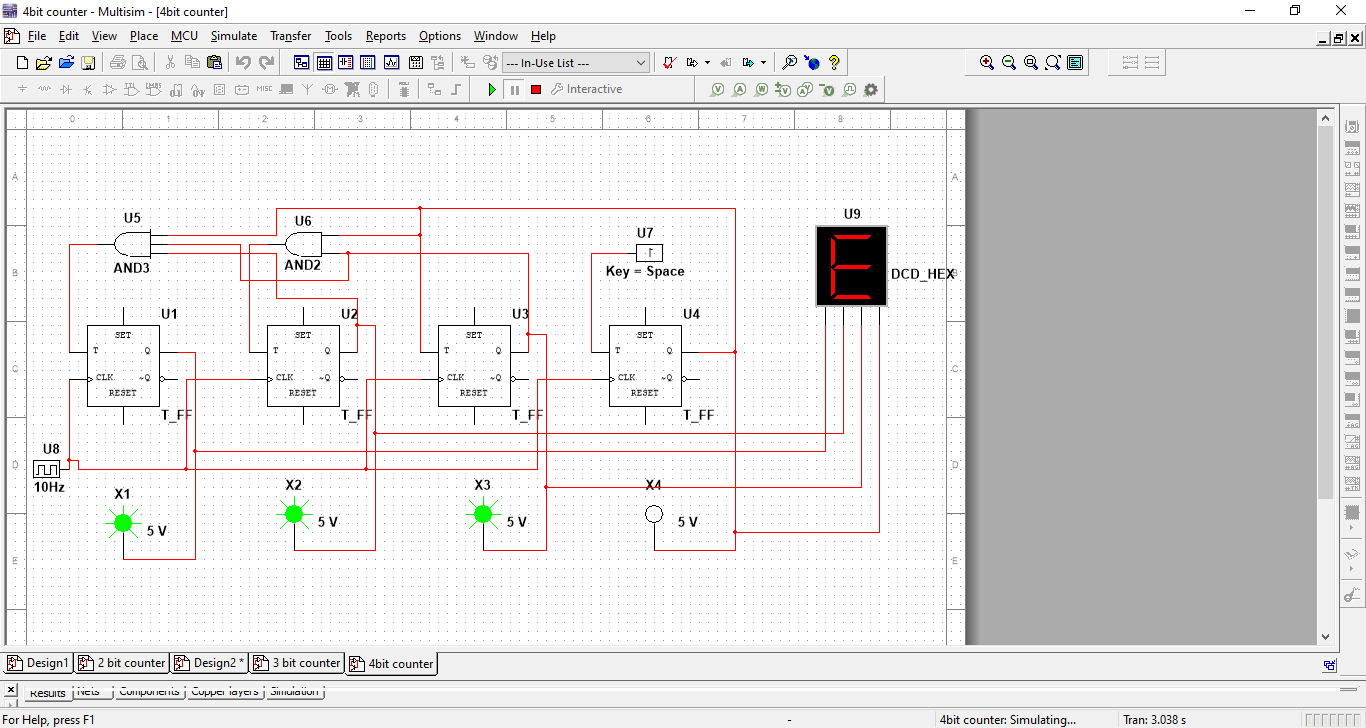
**Circuit Diagram(up counter):**

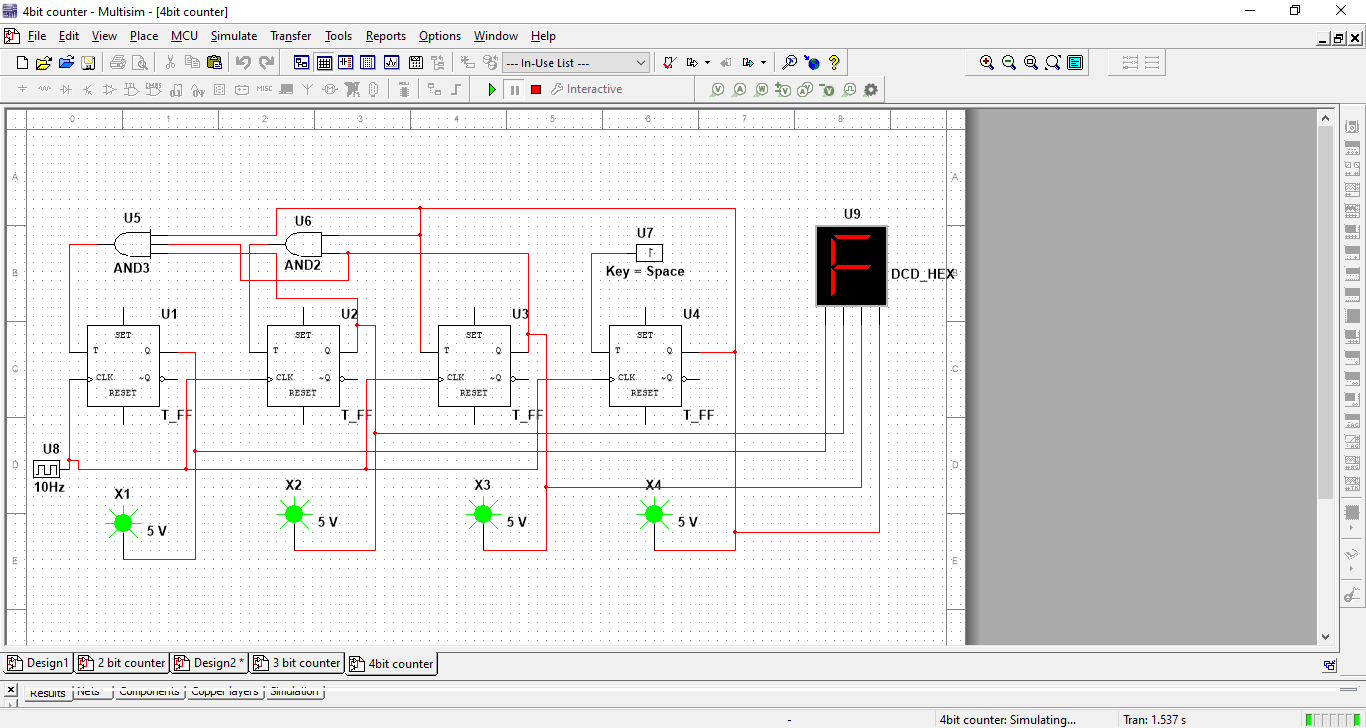


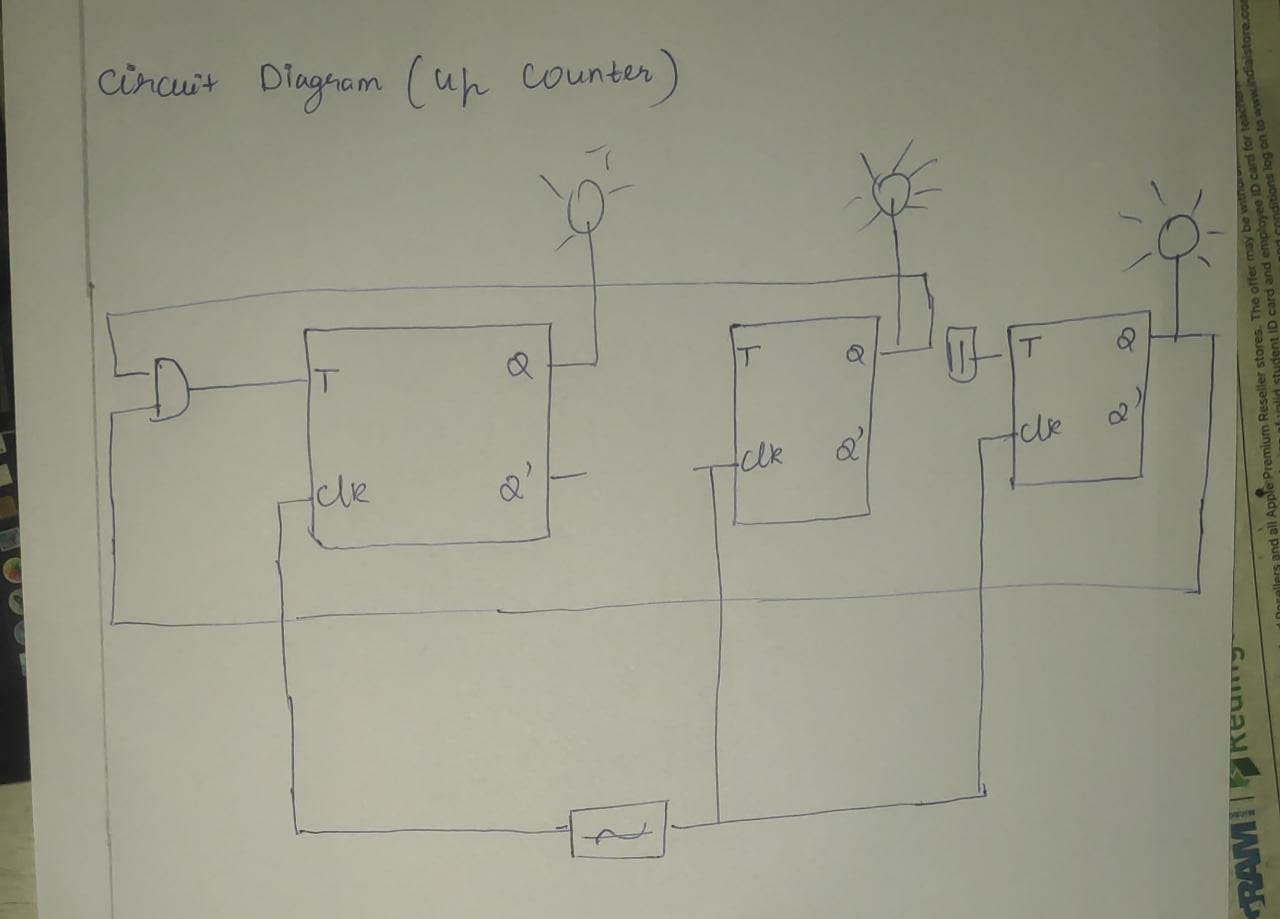
**Sample Output:**

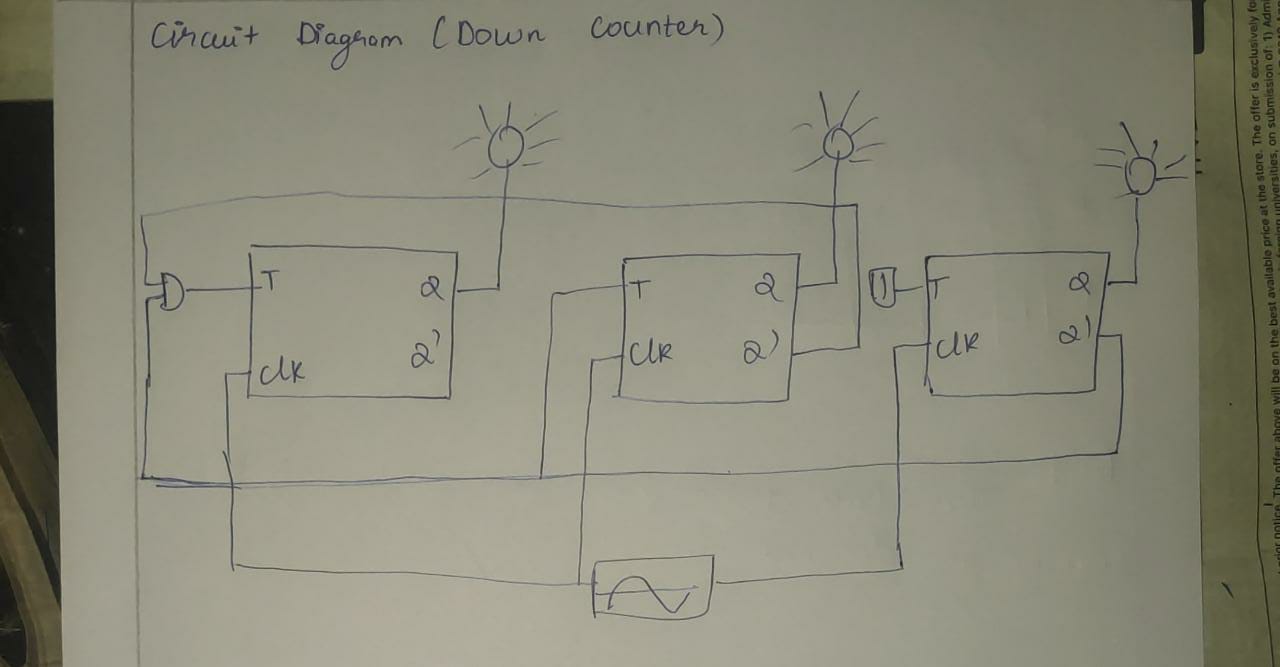






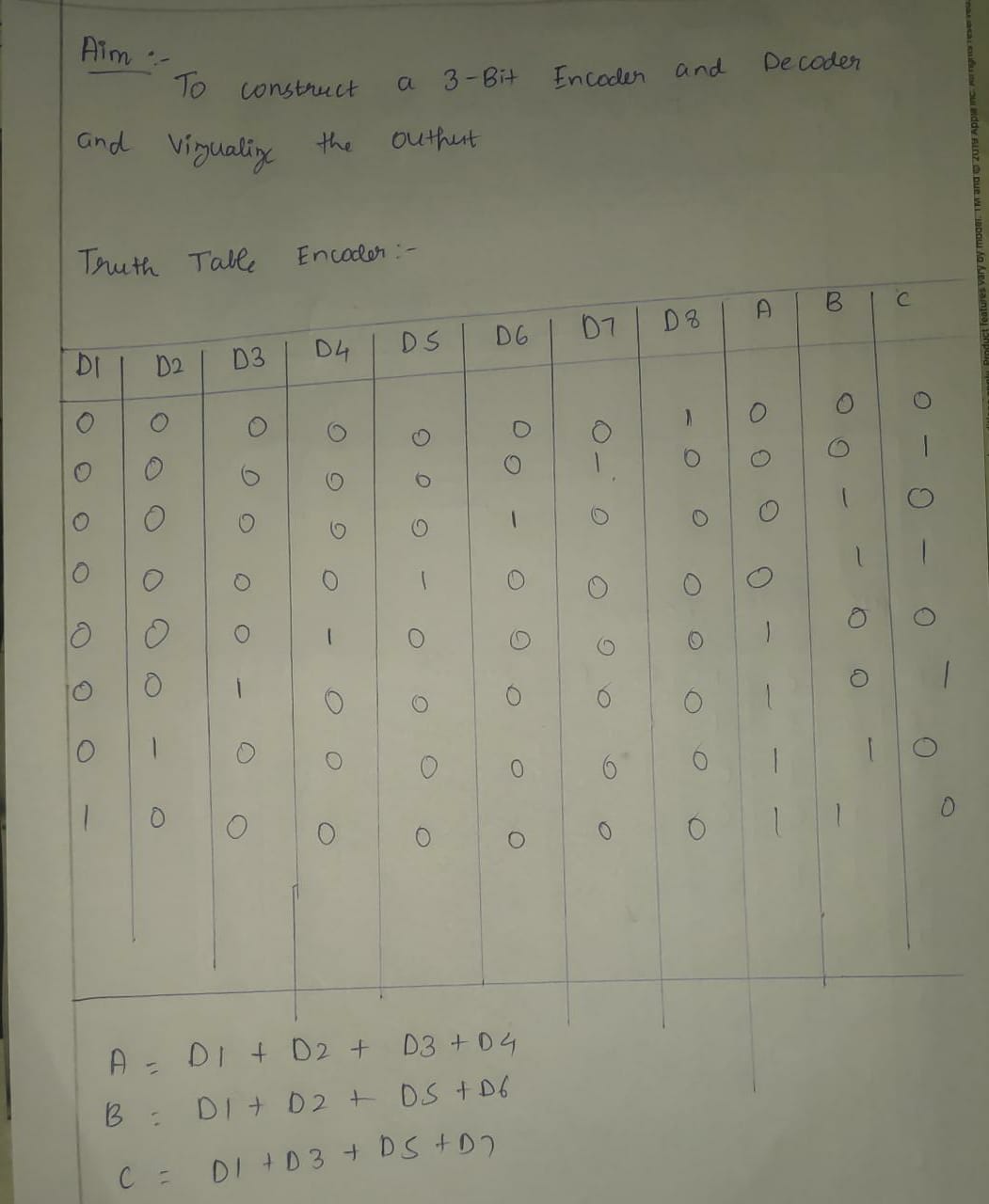


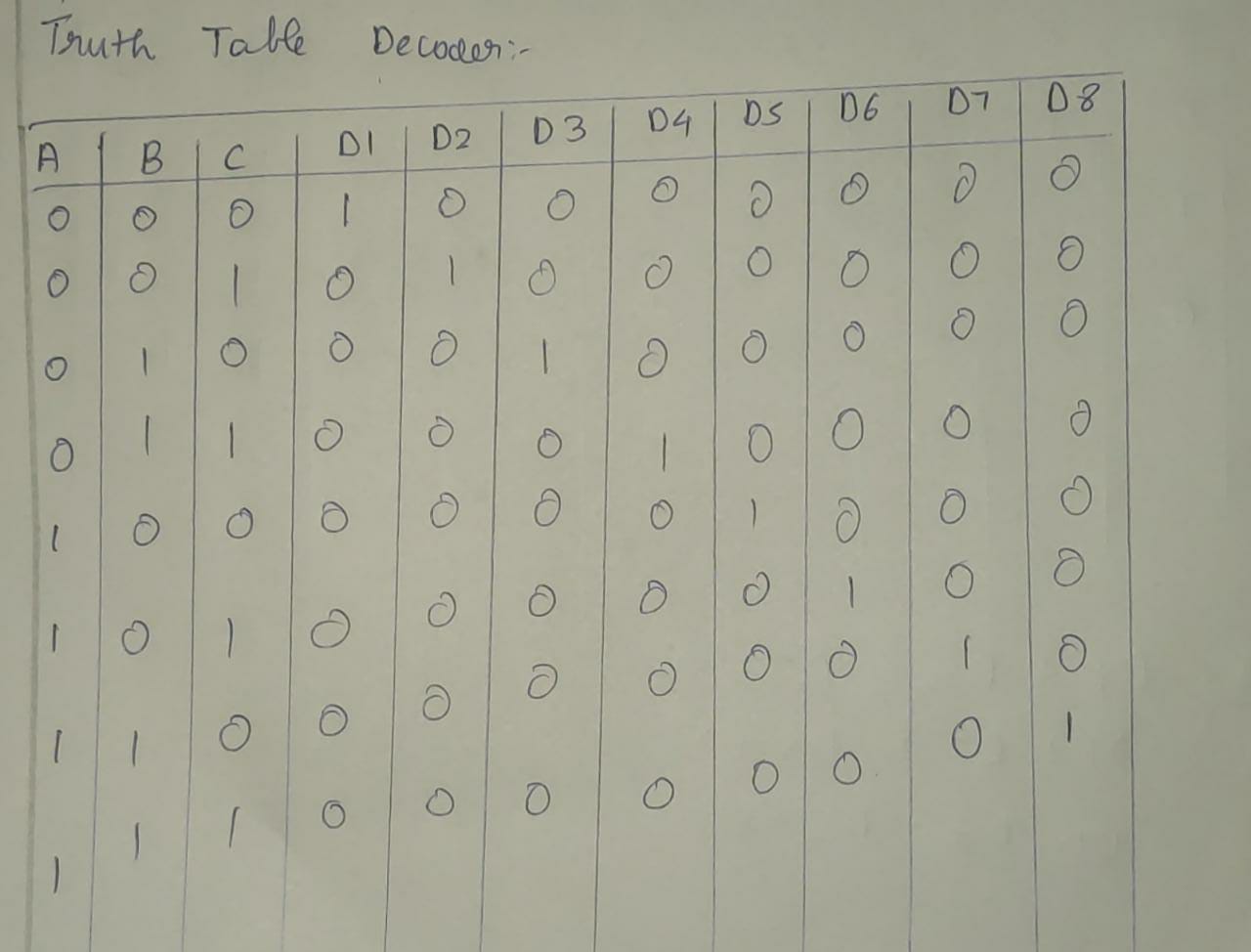
****

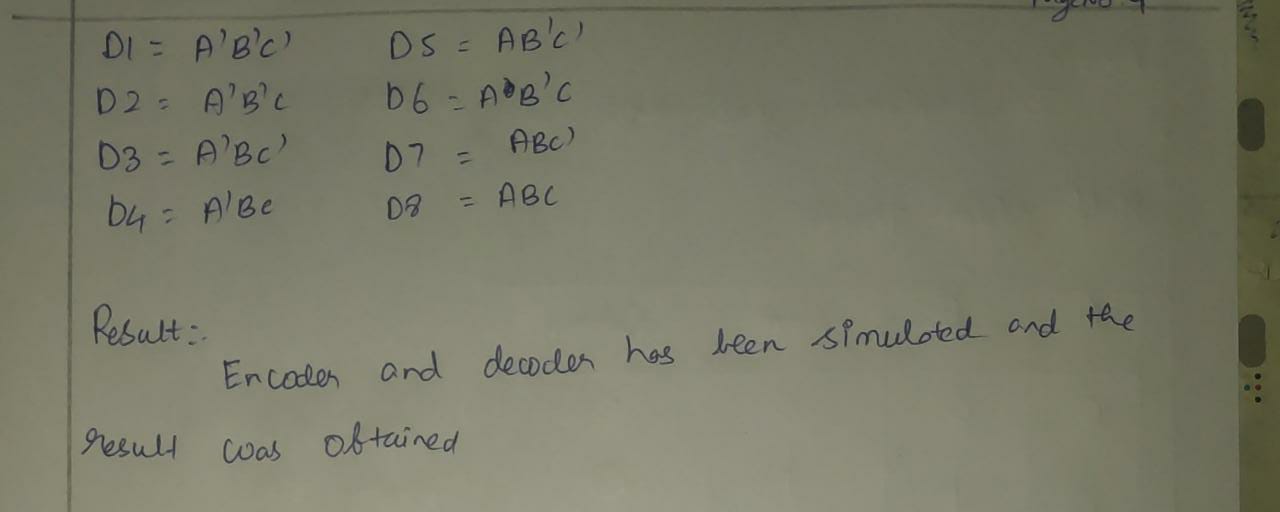
****

**Experiment Number: 3**

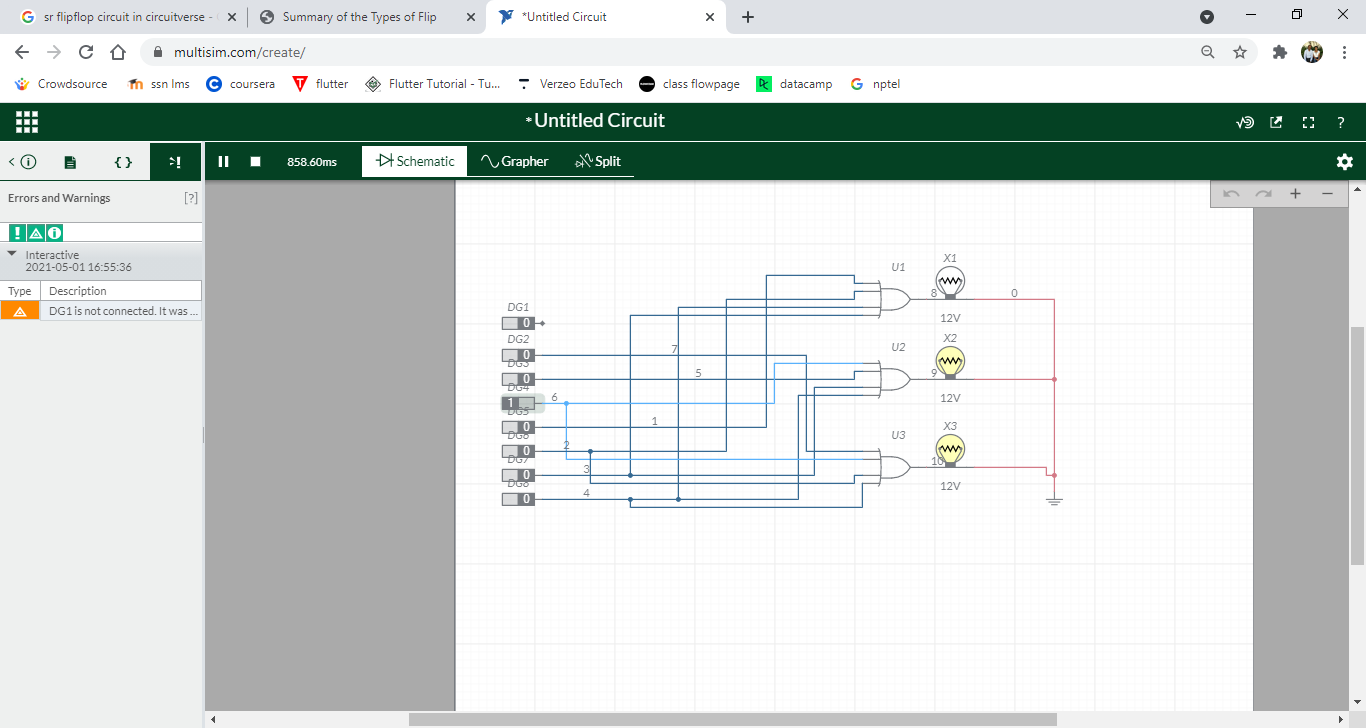
**Encoder and decoder**

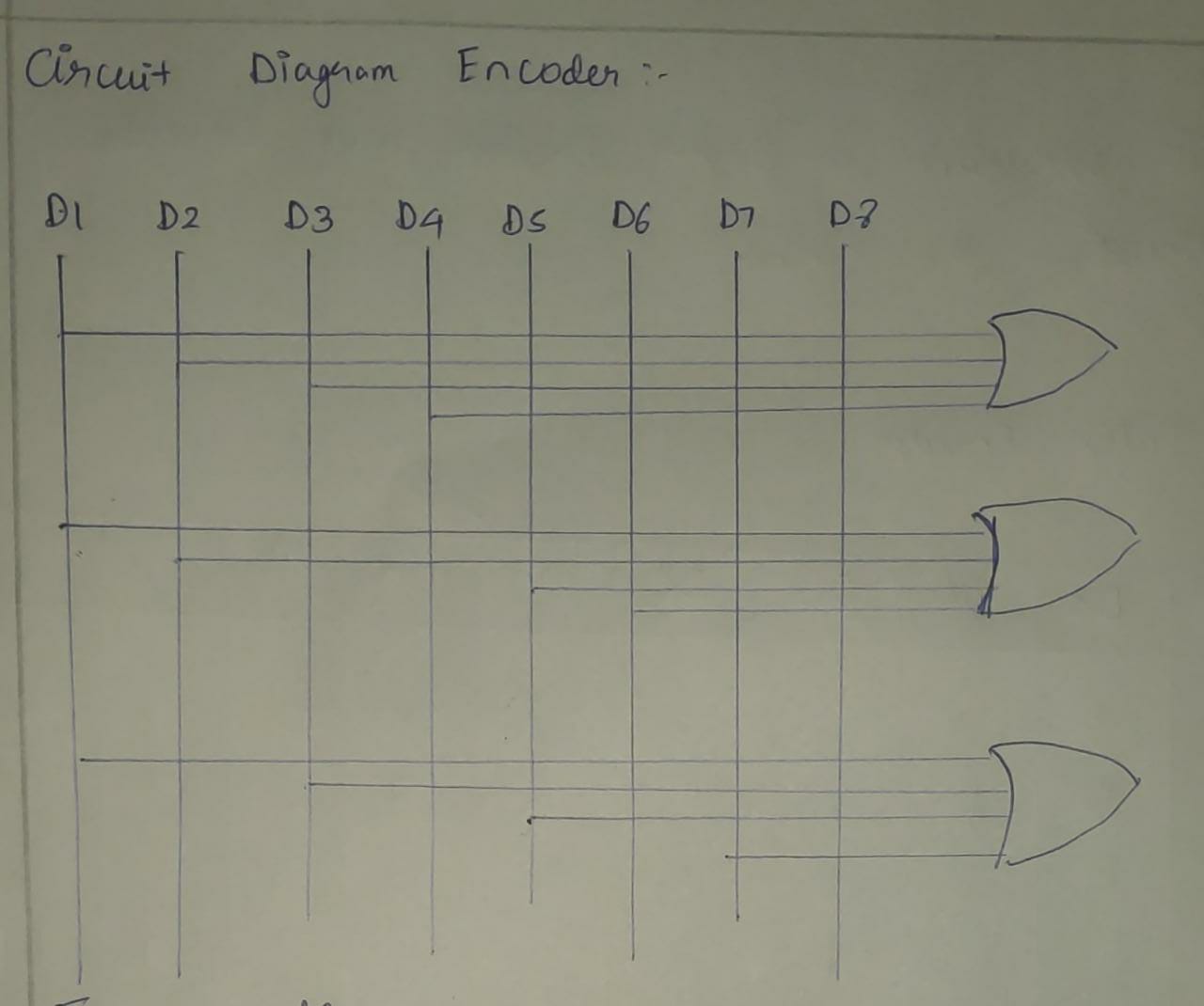
****

****

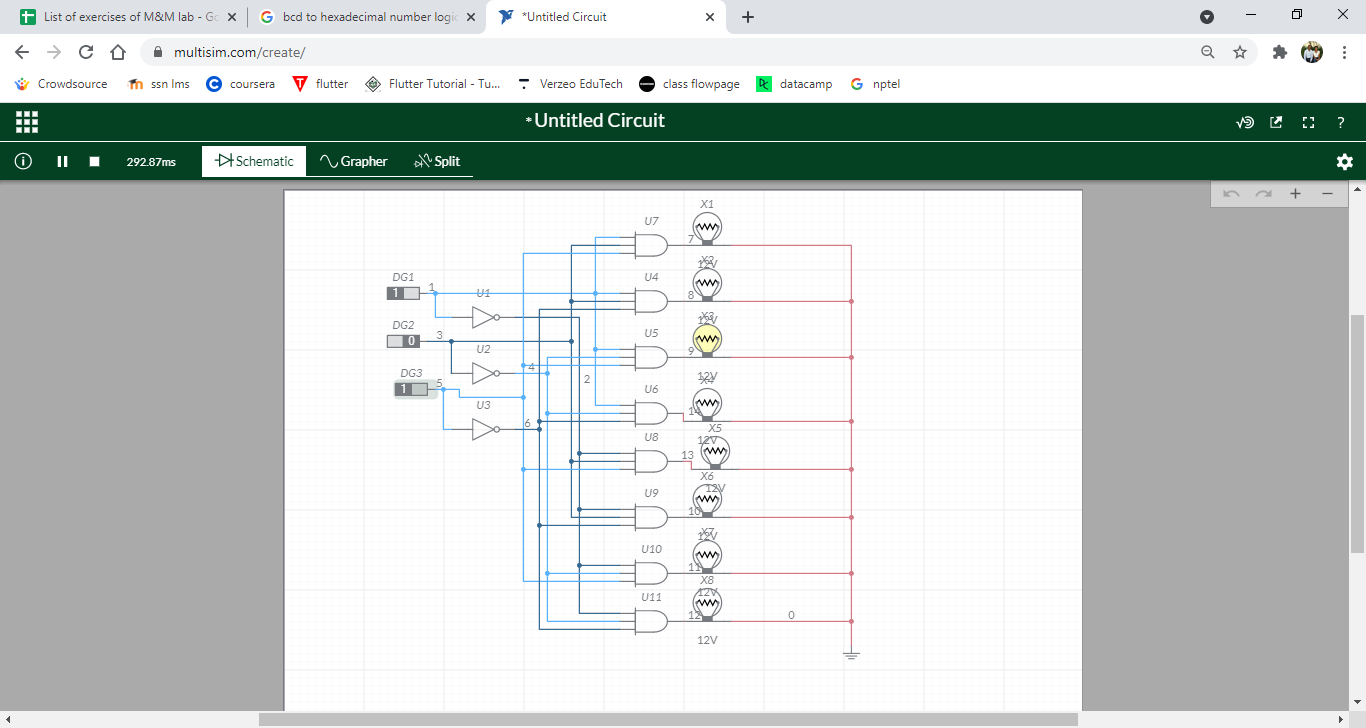
****

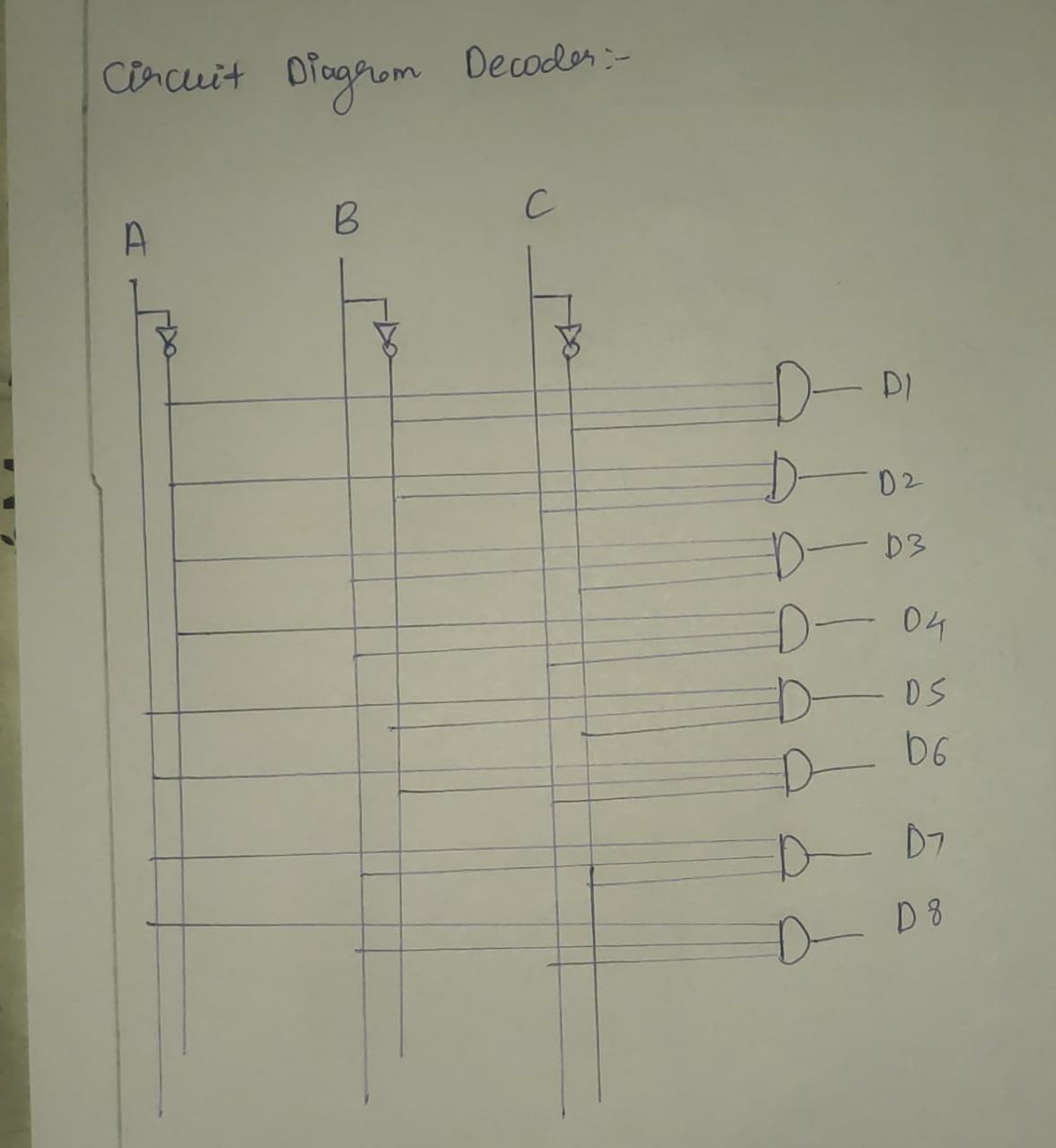
**Circuit Diagram:**



****

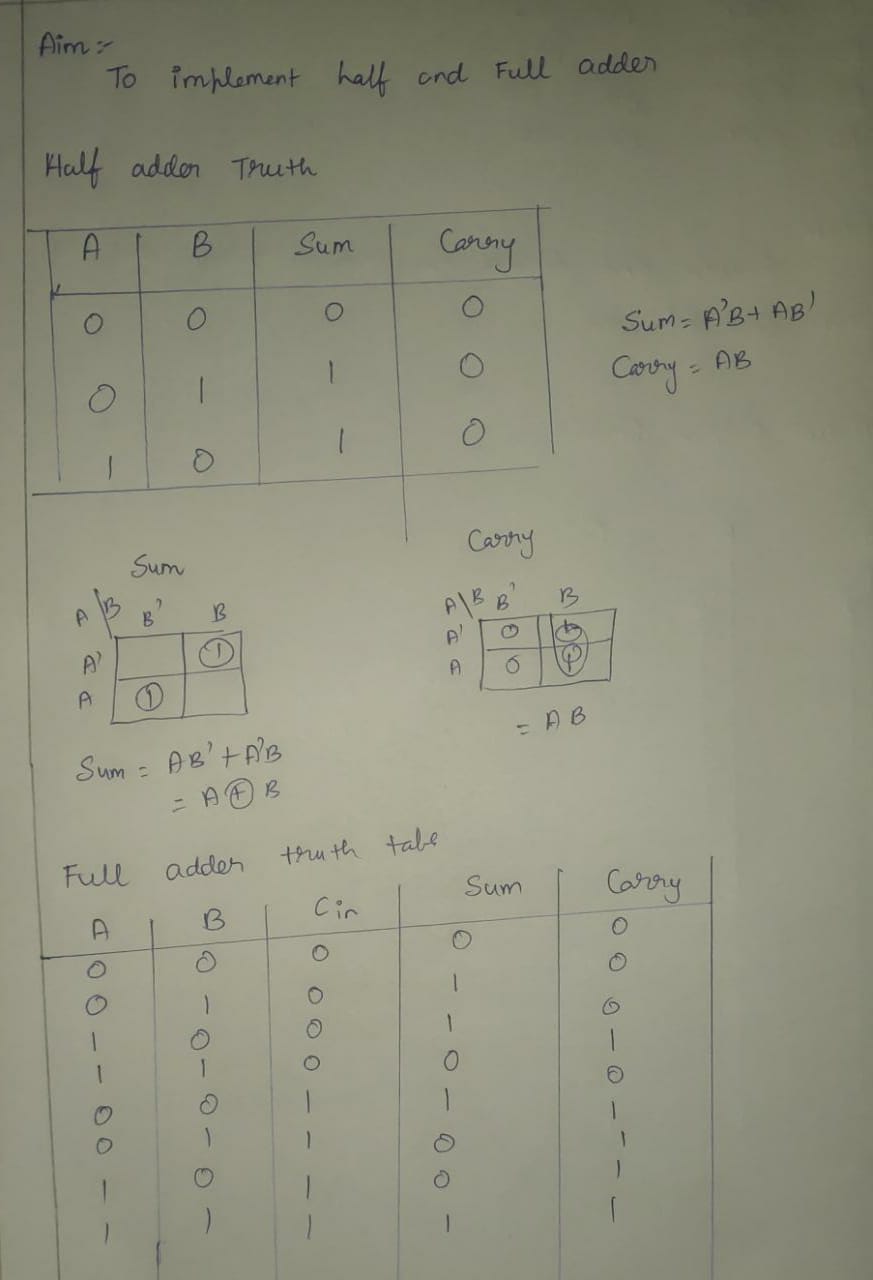
**Circuit Diagram(Decoder)**

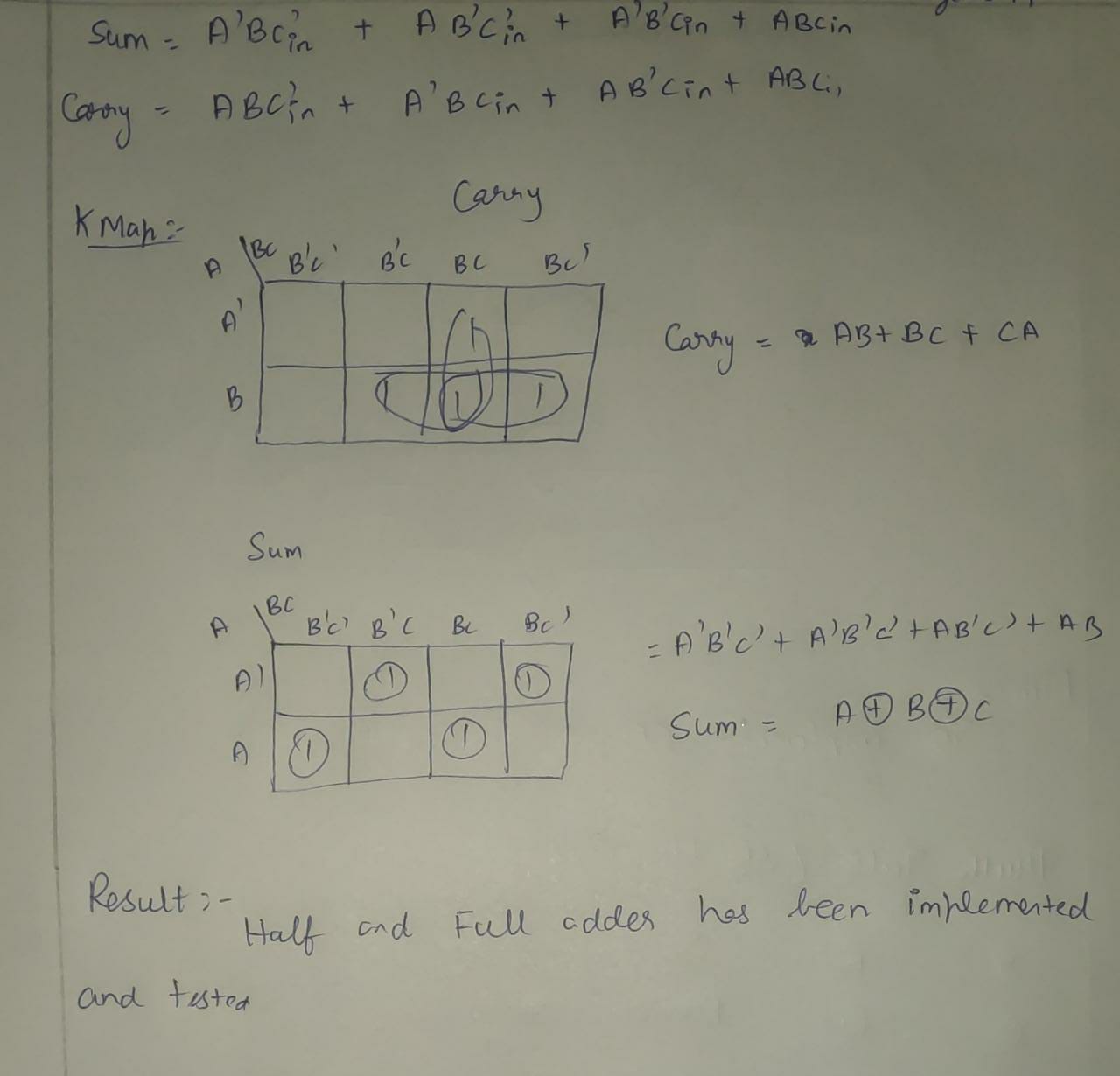


****

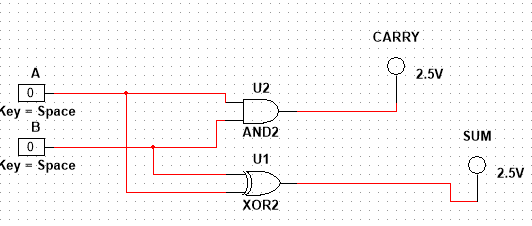
**Experiment Nnumber: 4**

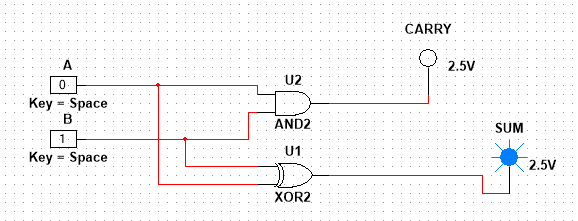
**Half adder and full adder**

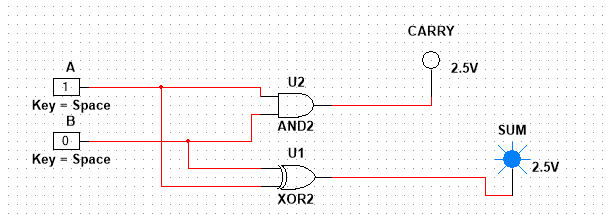
****

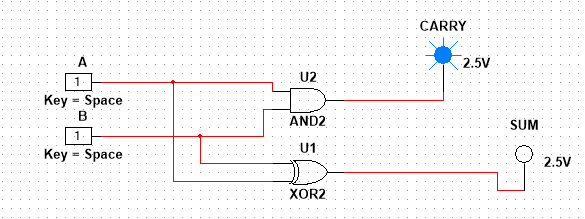
****

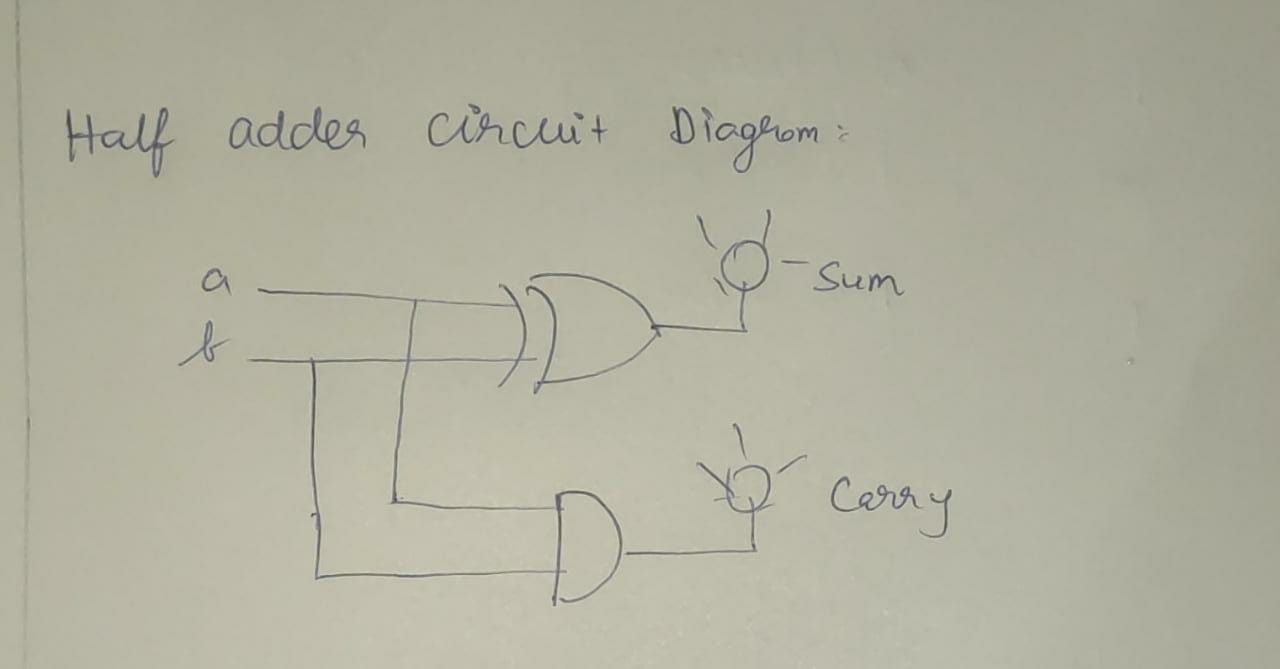
**Circuit Diagram:**



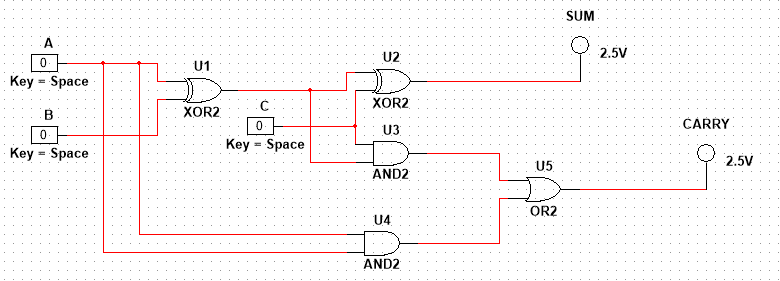


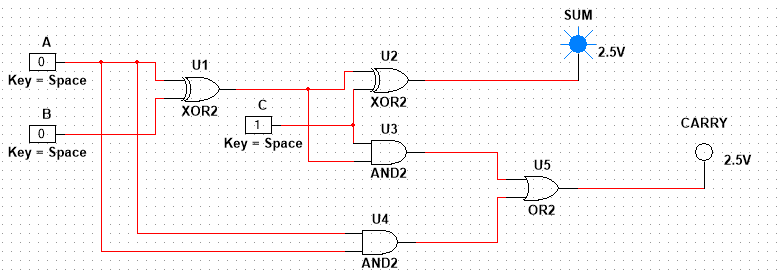


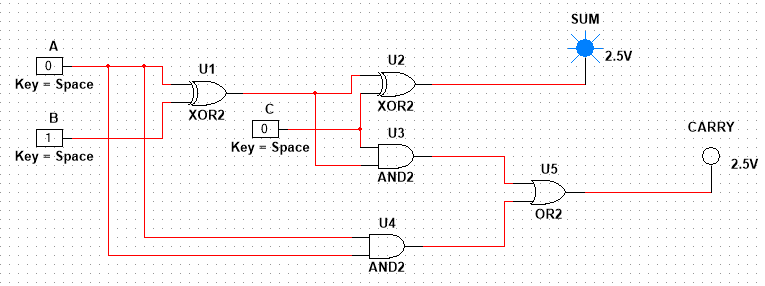


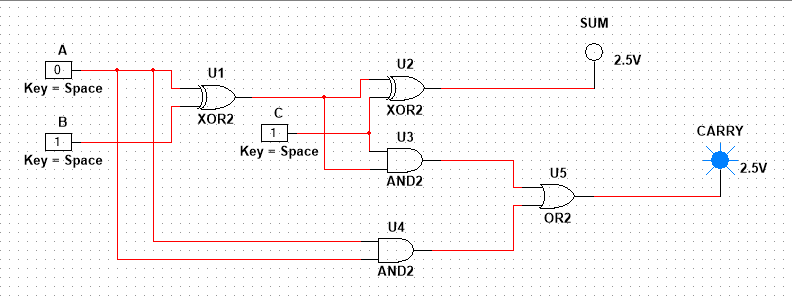
****

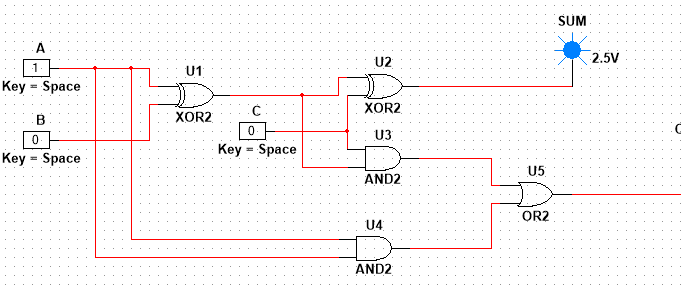
**Circuit Diagram(Full-Adder)**



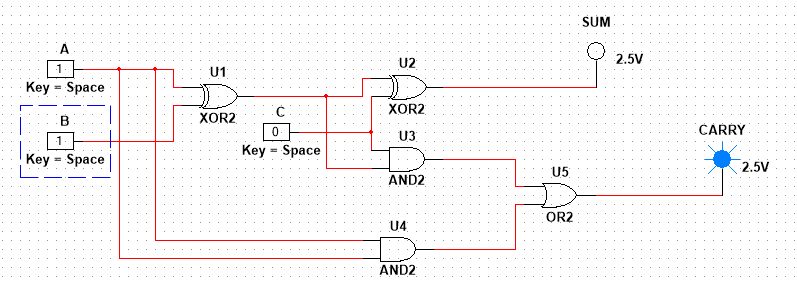


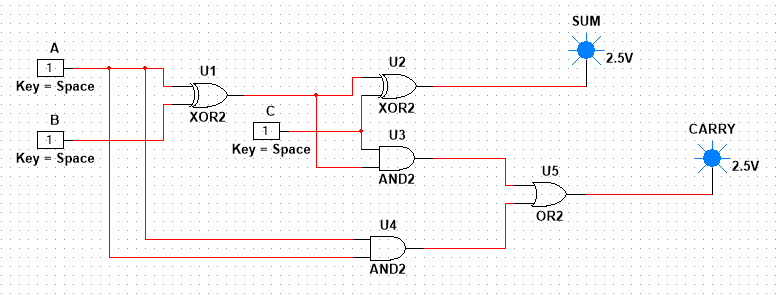


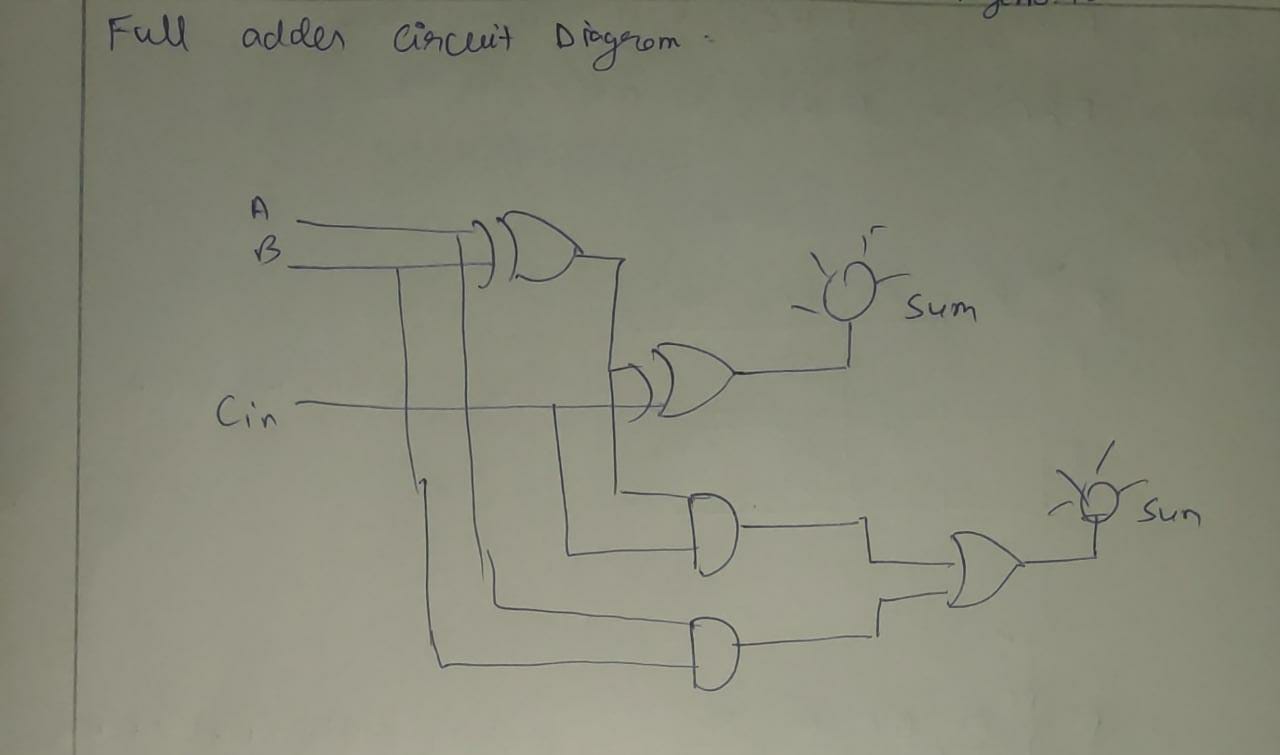






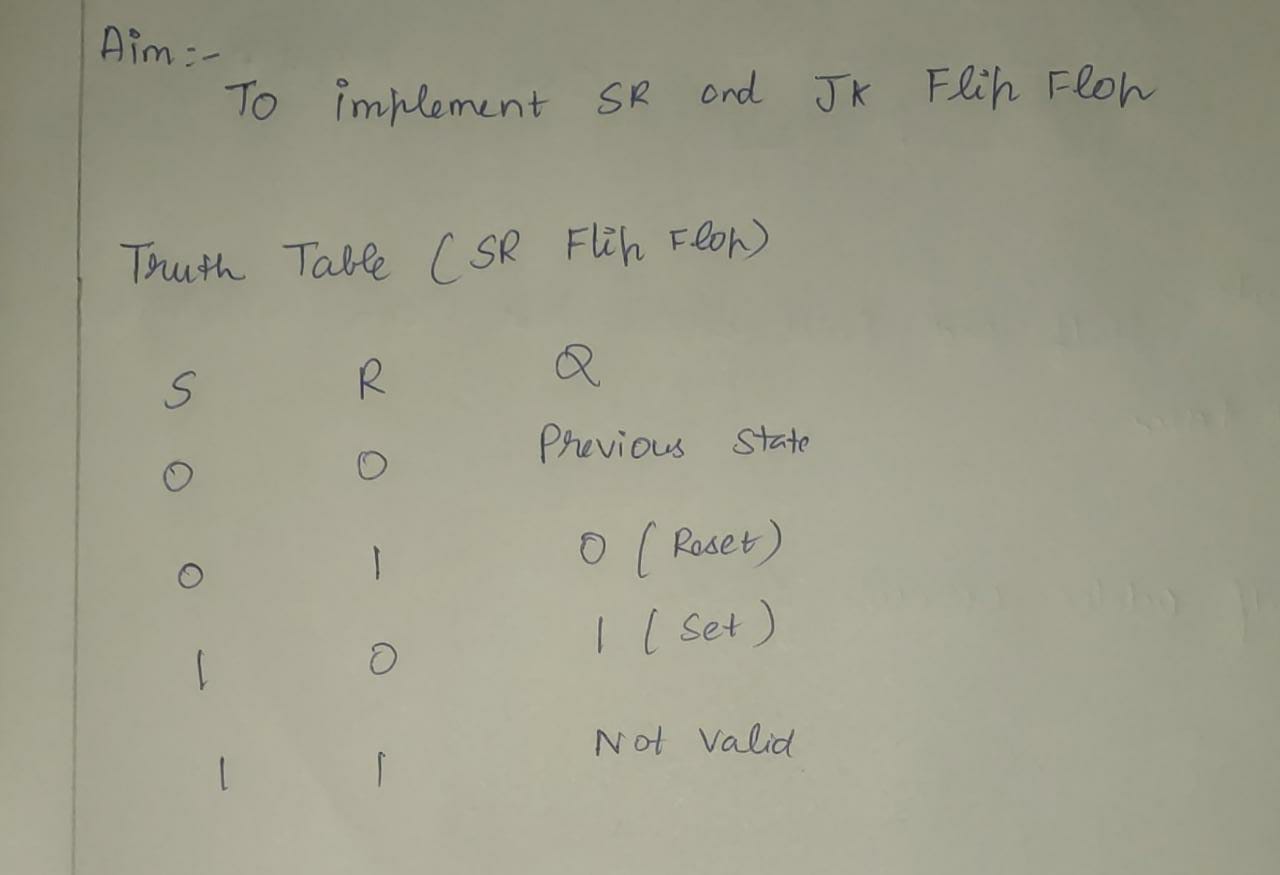


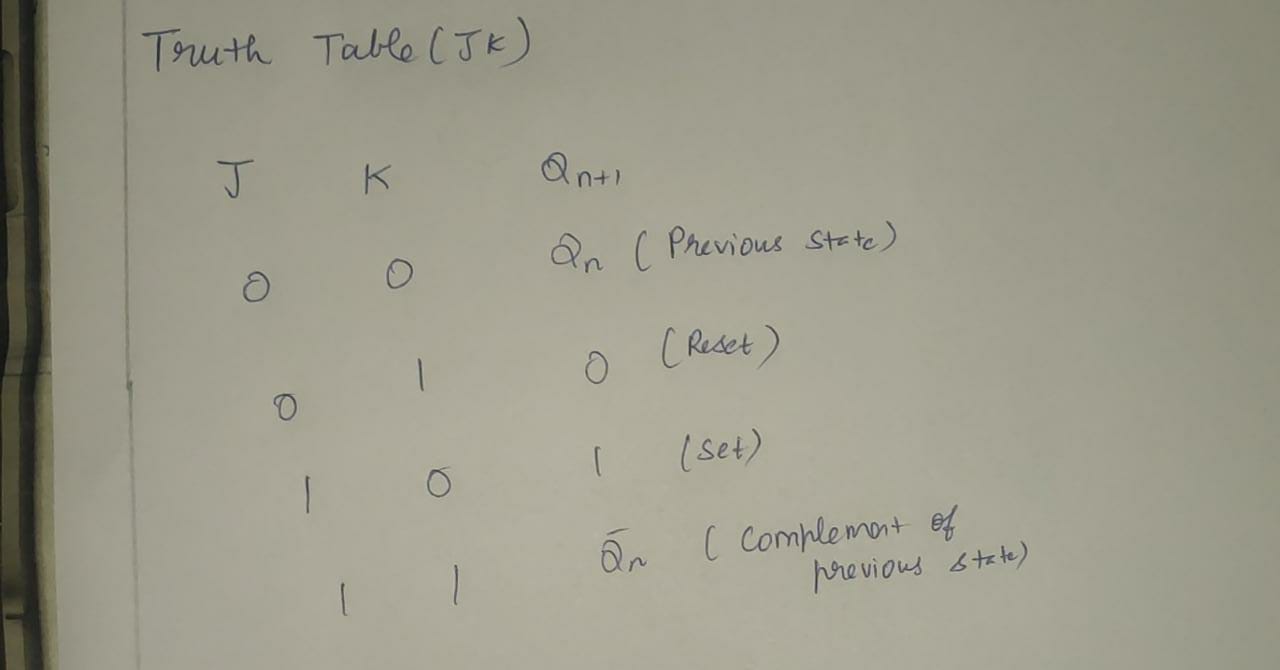


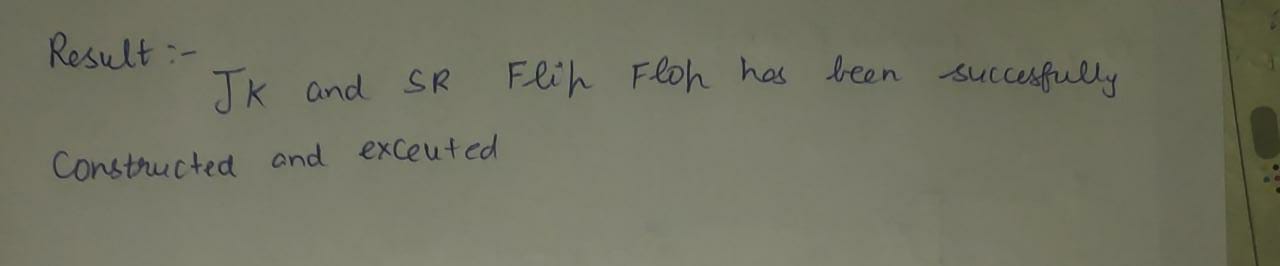
****

**Experiment Number: 5**

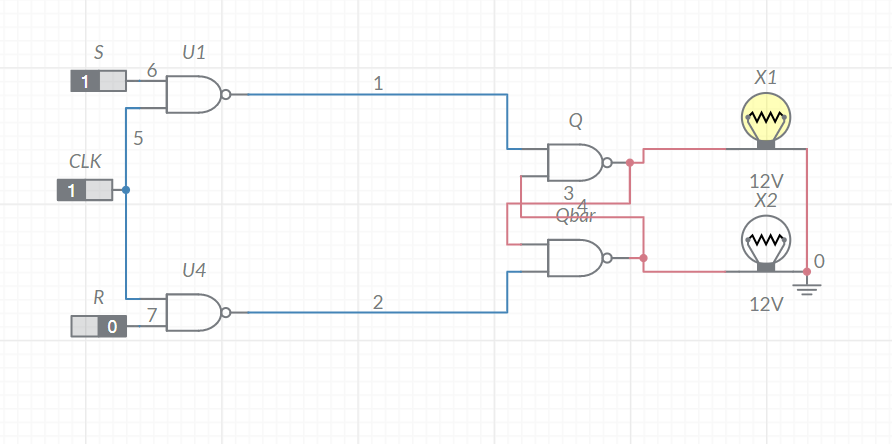
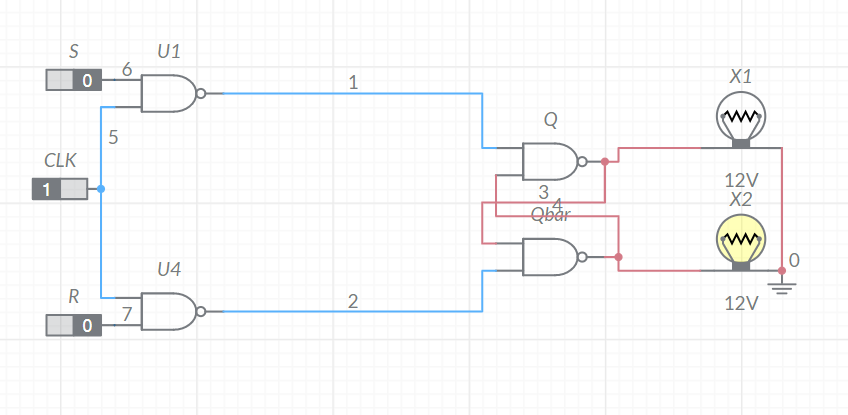
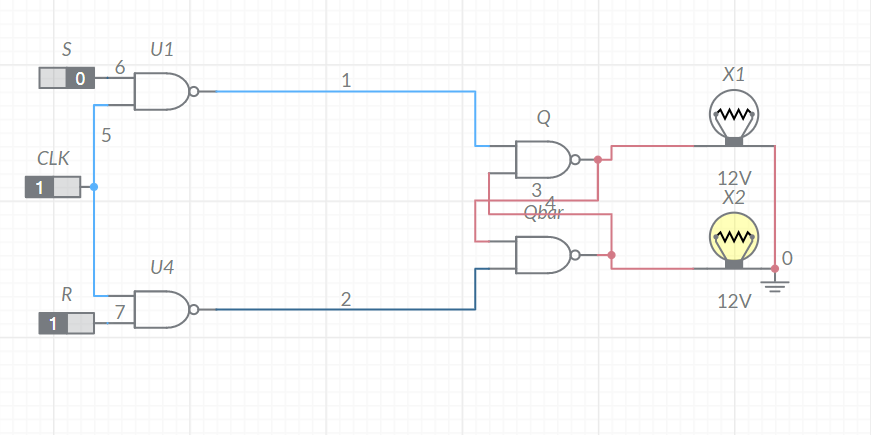
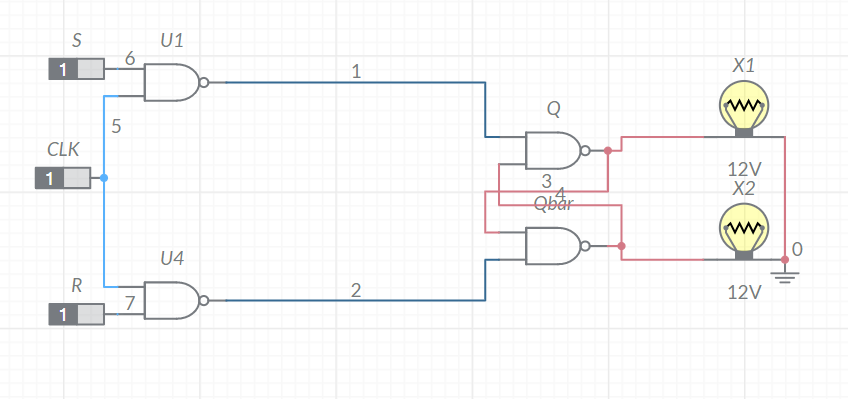
**JK and SR flip flop**

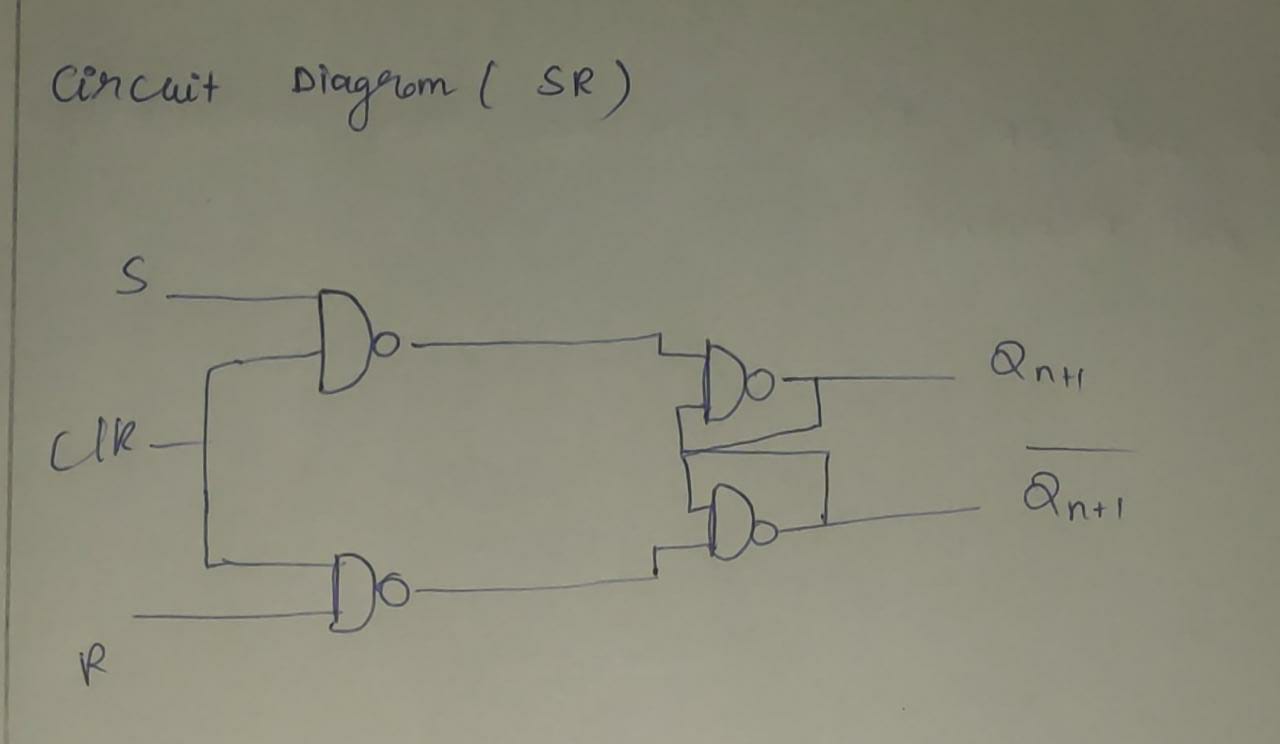
****

****

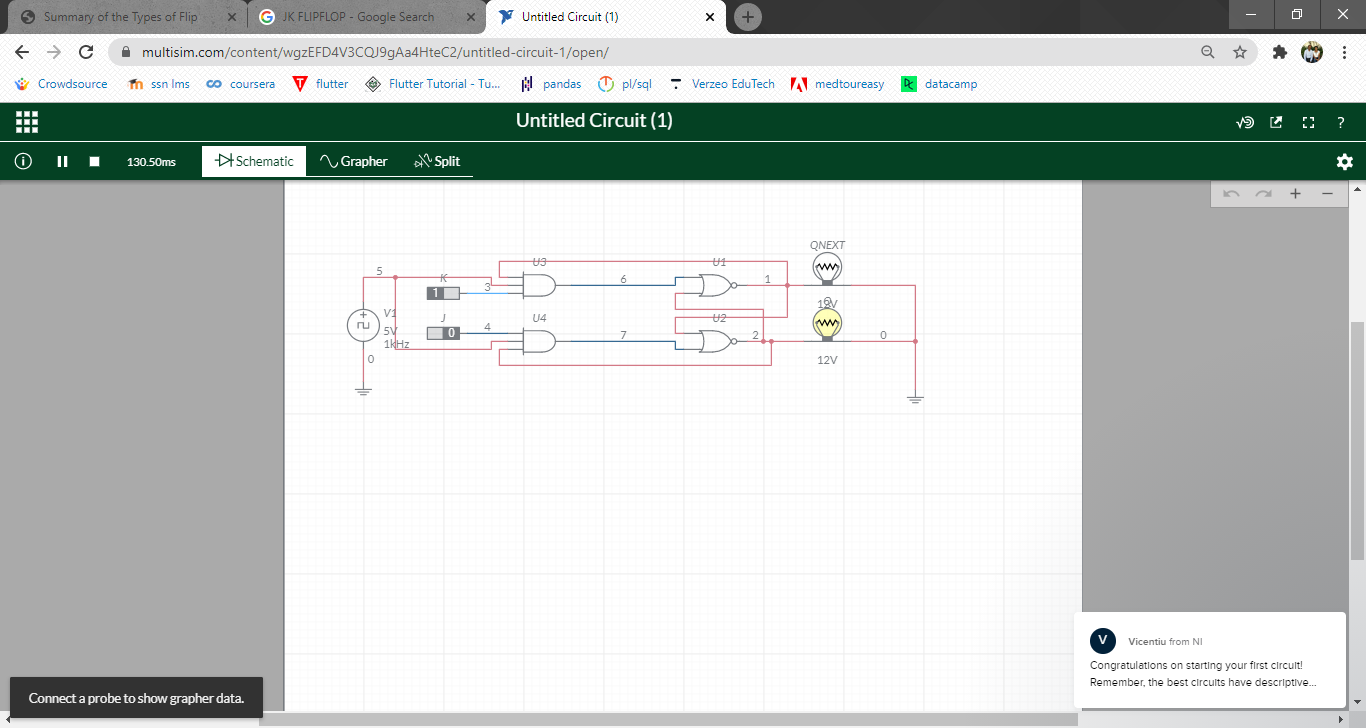
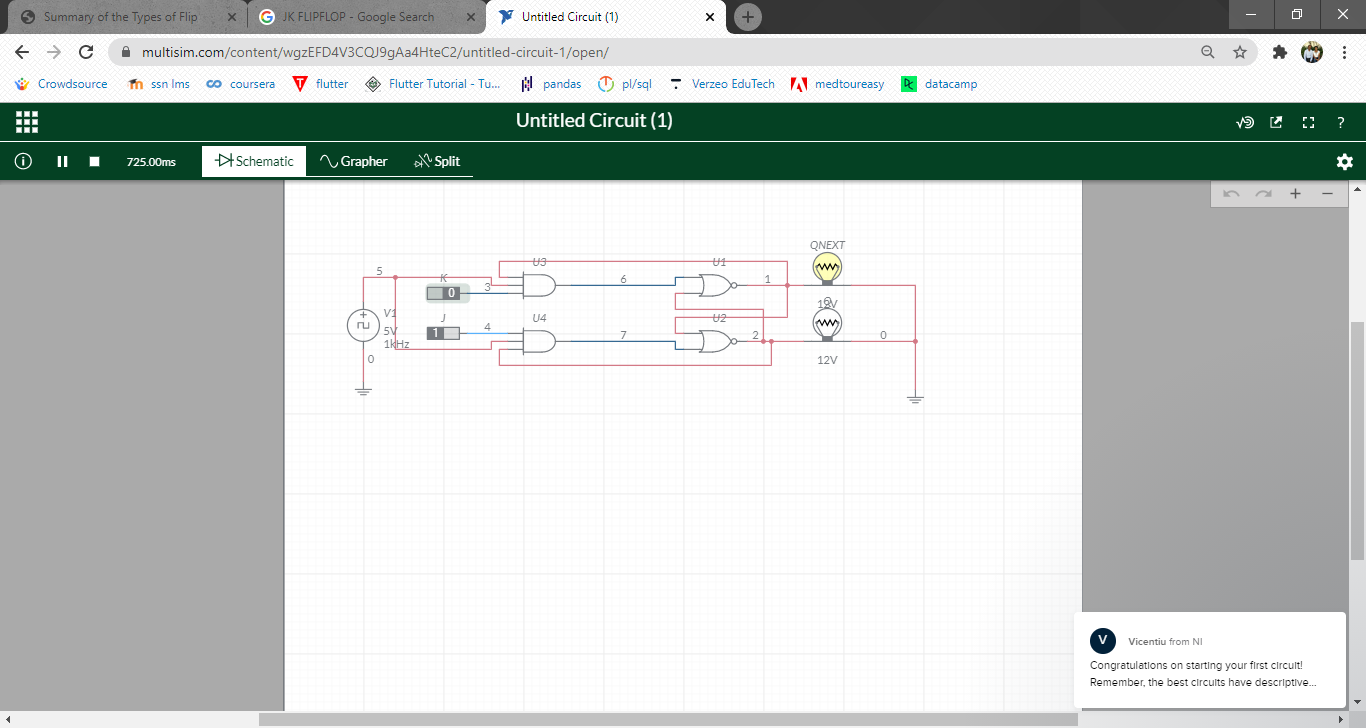
****

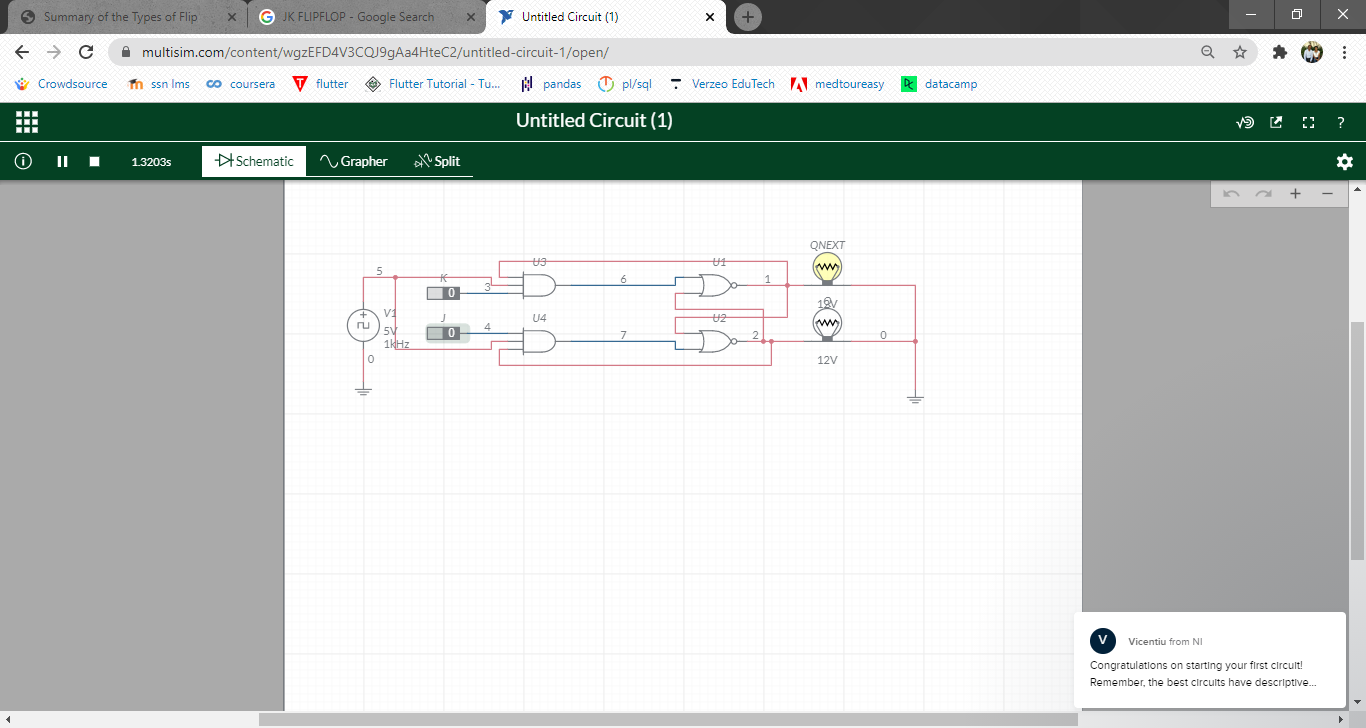
**Circuit Diagram:**

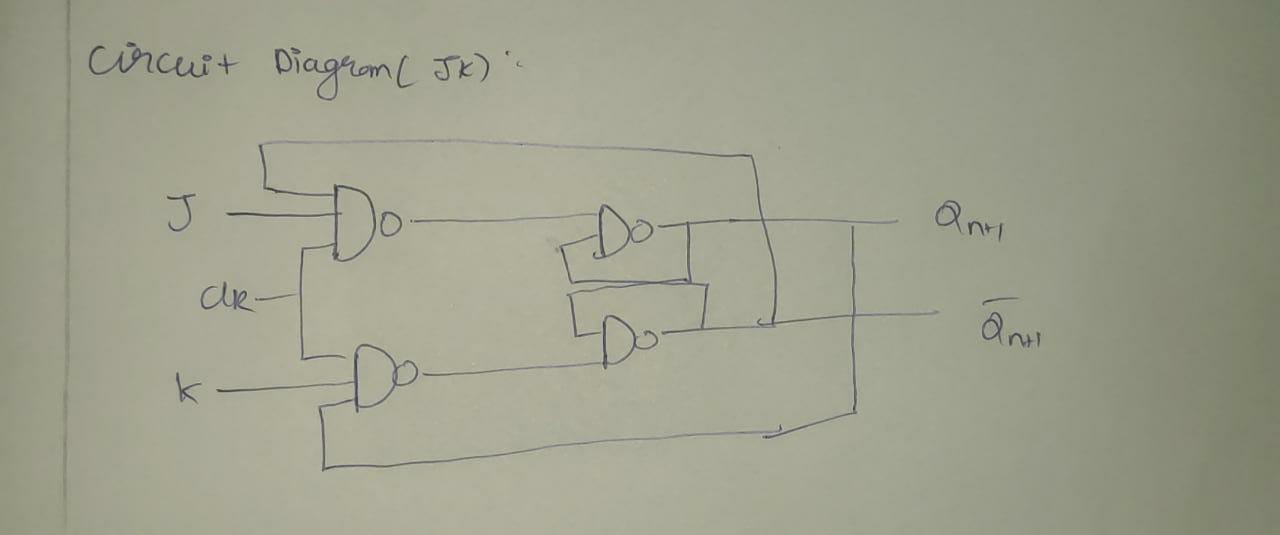
  

****

**Circuit Diagram(JF flip flop)**

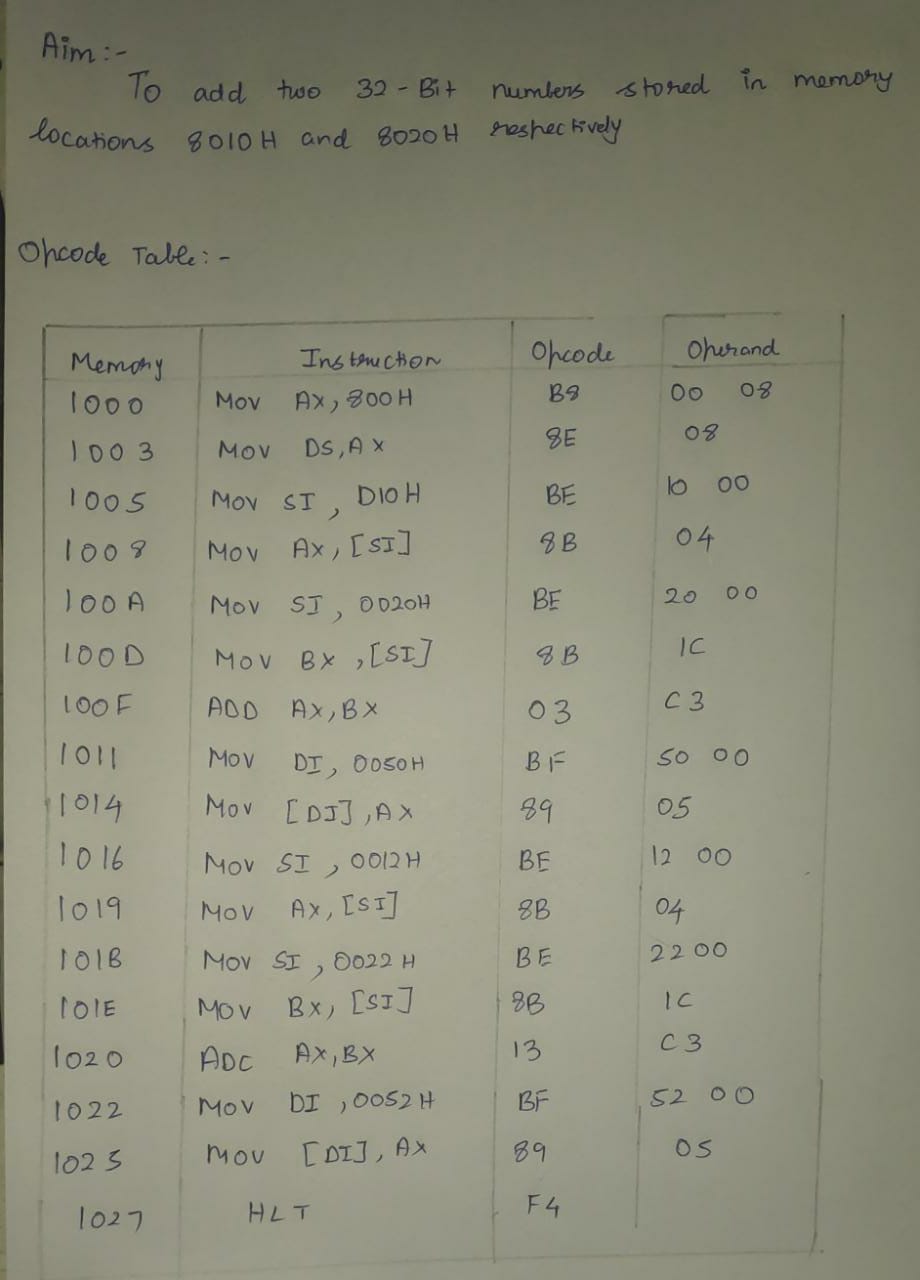




****

**ASSEMBLY LANGUAGE PROGRAMMING WITH 8086**

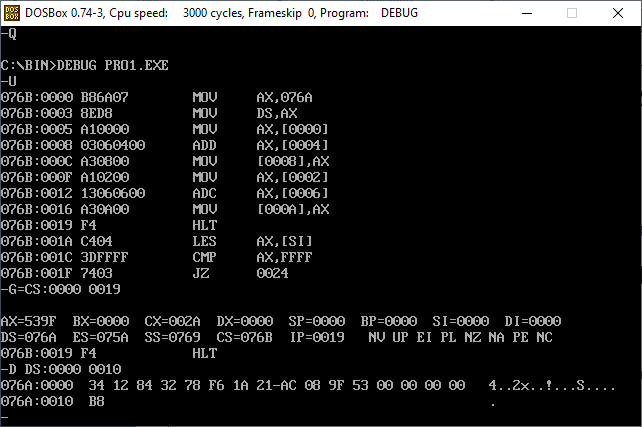
**Experiment Number-1 To Add two 32-Bit Number**

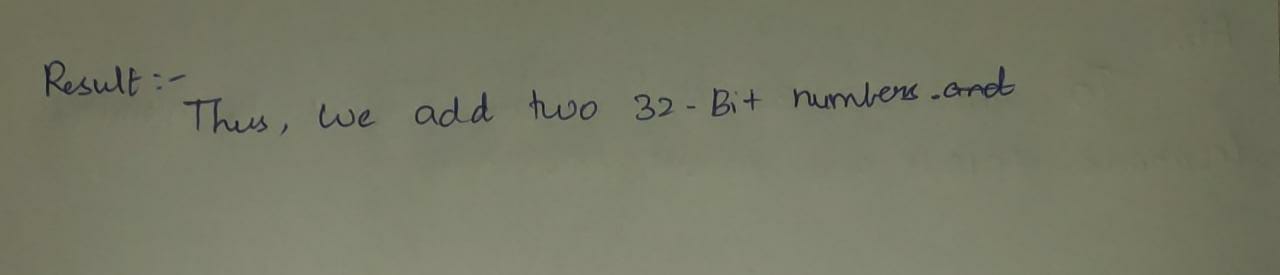
****

**Code:**



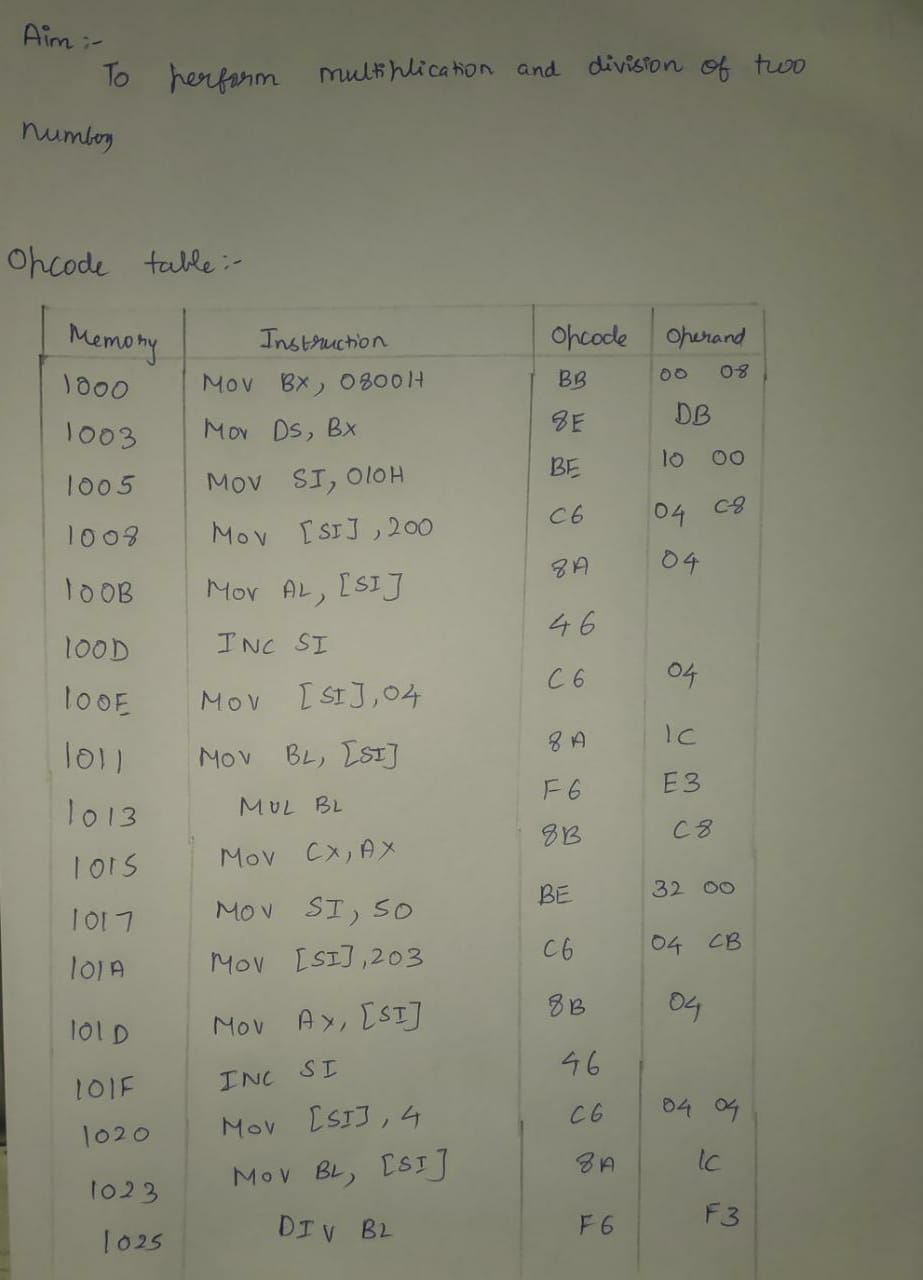
**Output:**



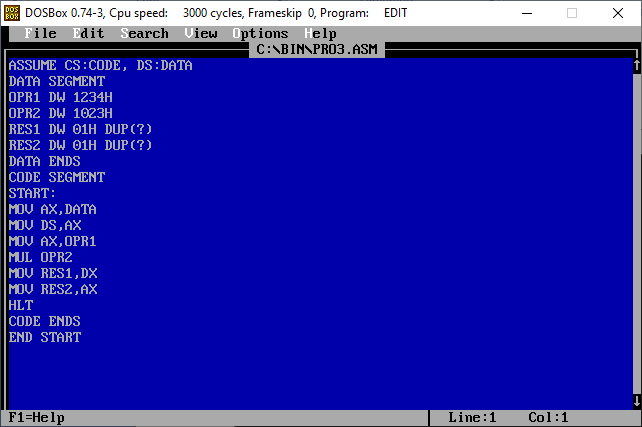
****

**Experiment Number-2**

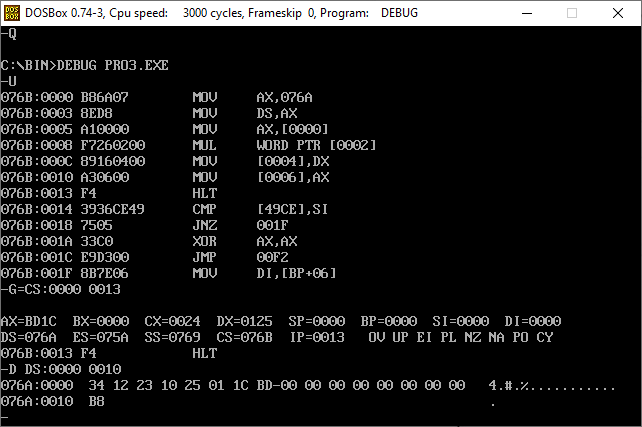
**To Perform Multiplication and Division of Two Numbers**

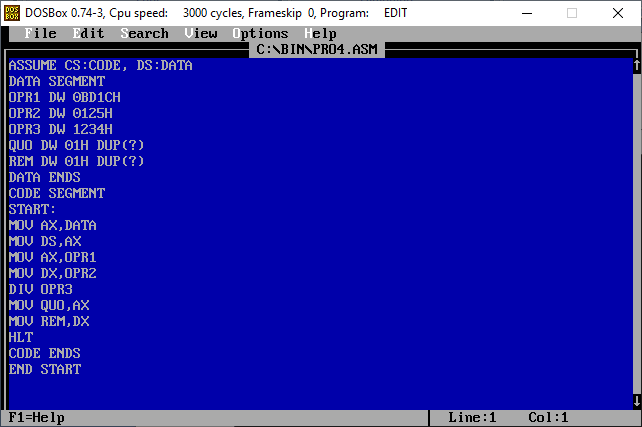
****

**Code(multiplication):**

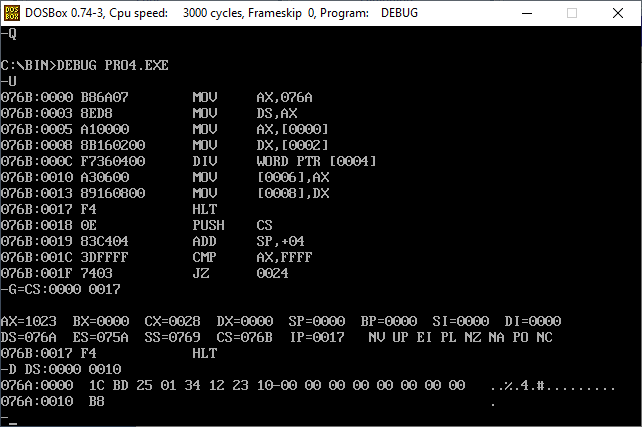


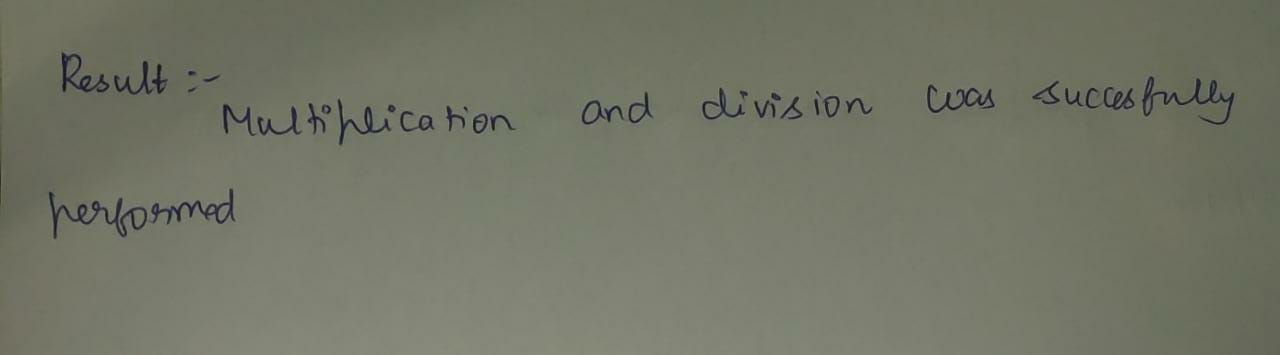
**Output:**



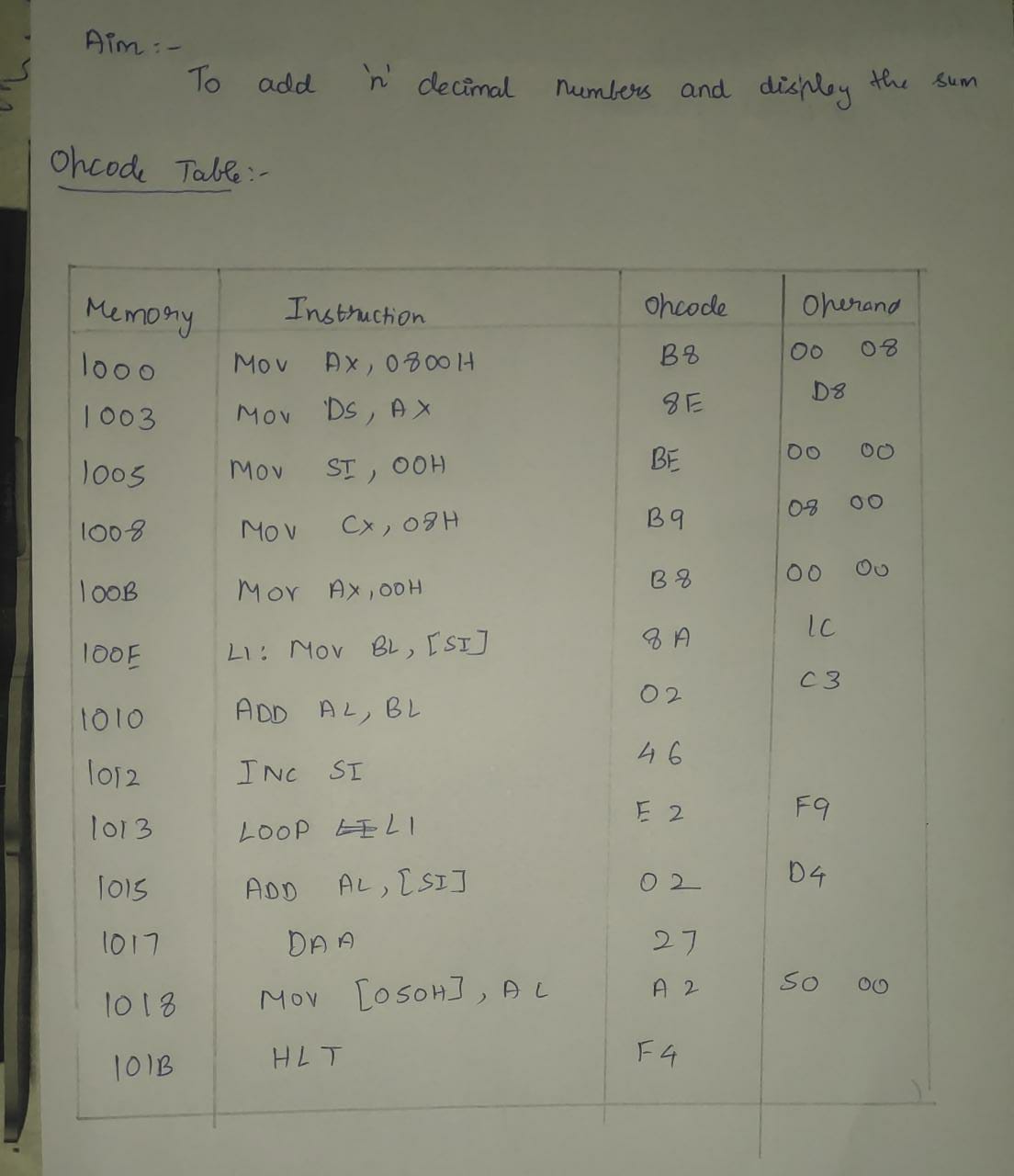
**Code(division):**

**Output:**

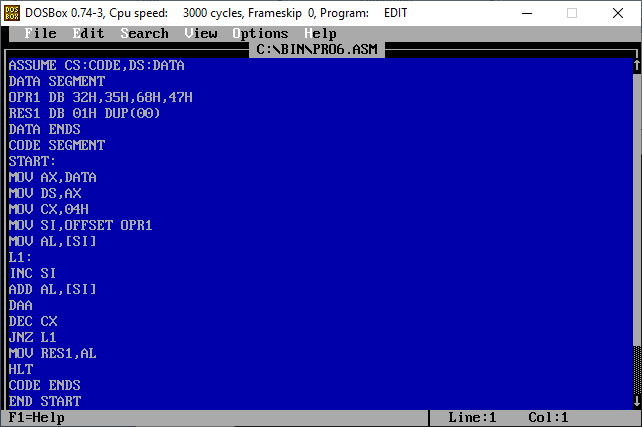


****

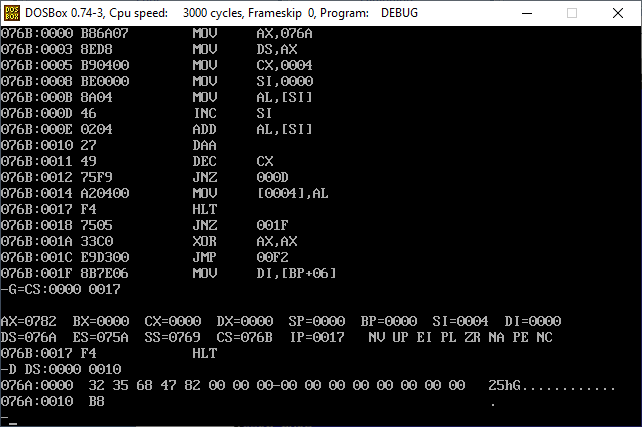
**Experiment Number-3 To Add ‘N’ Decimal Numbers**

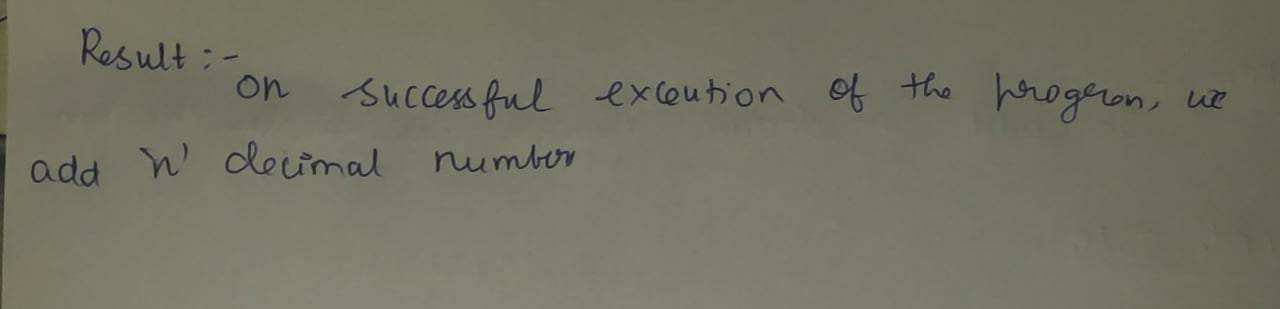
****

**Code:**



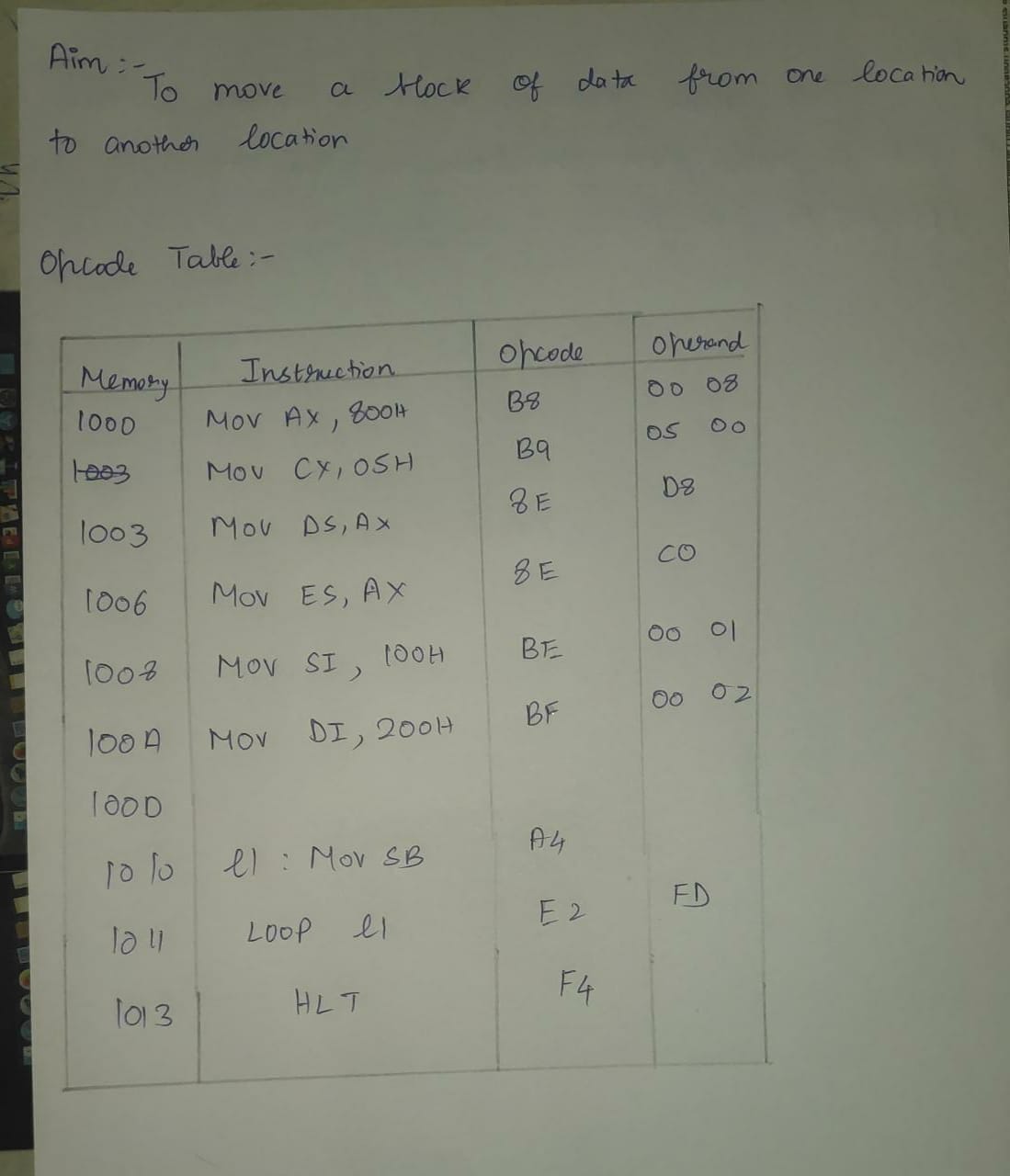
**Output:**



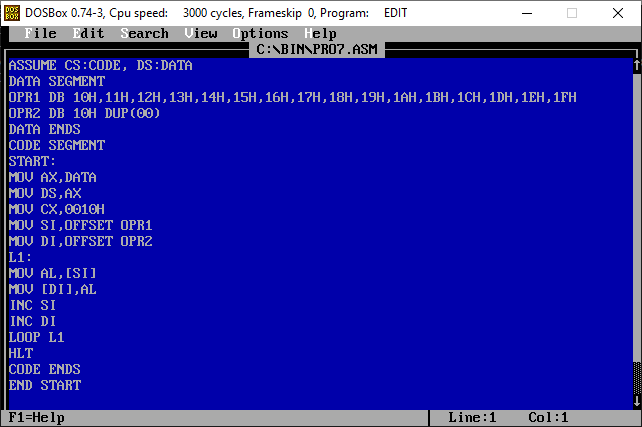
****

**Experiment Number-4**

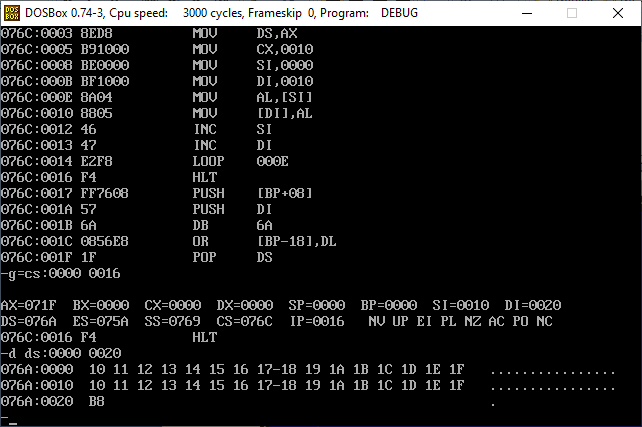
**To Move a Block of Data from one Location to Other**

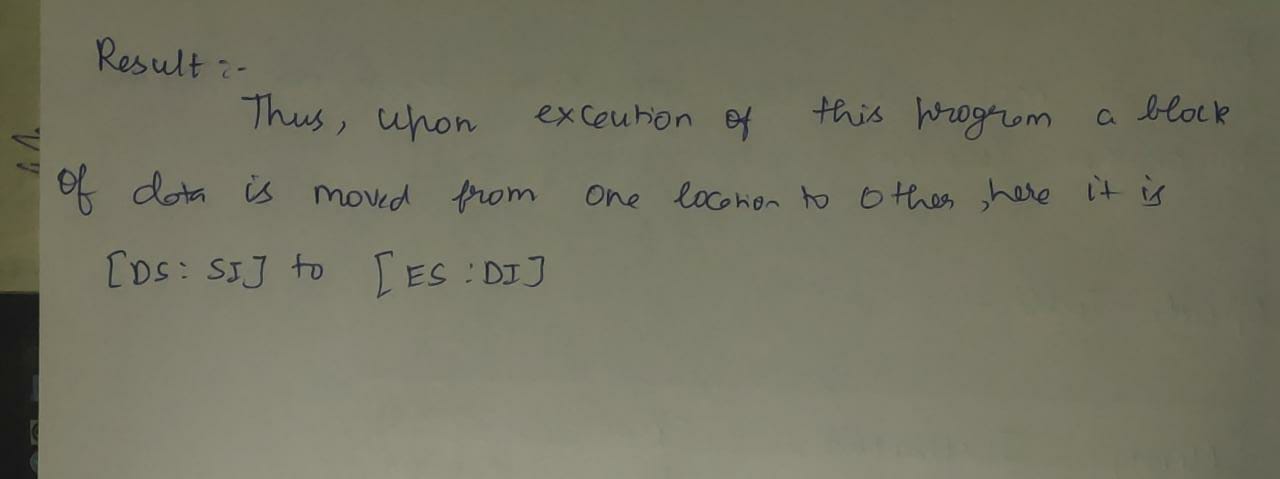
****

**Code:**

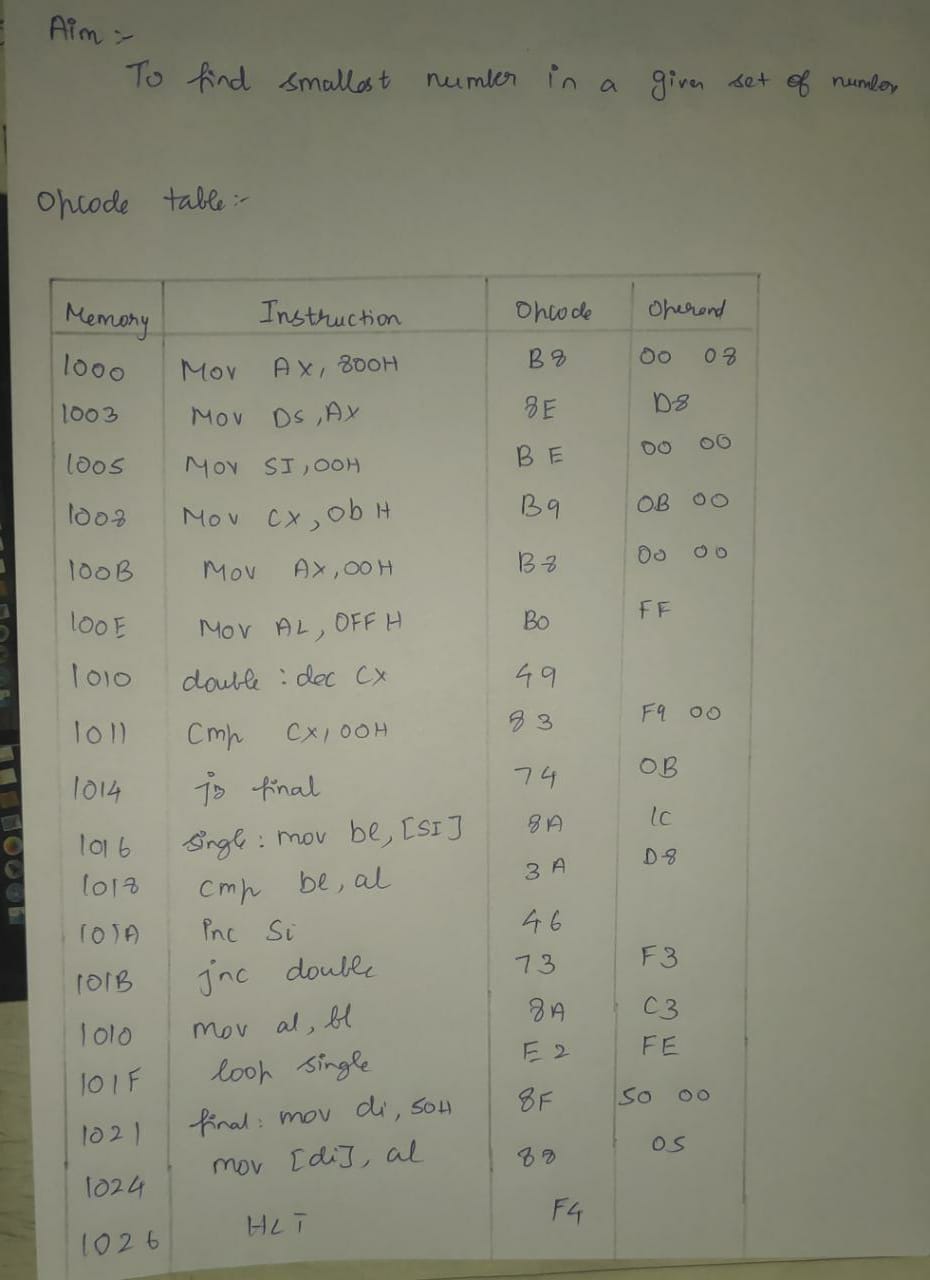


**Output:**

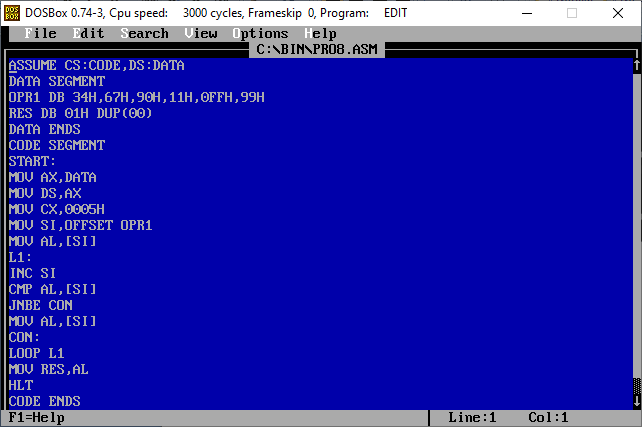


****

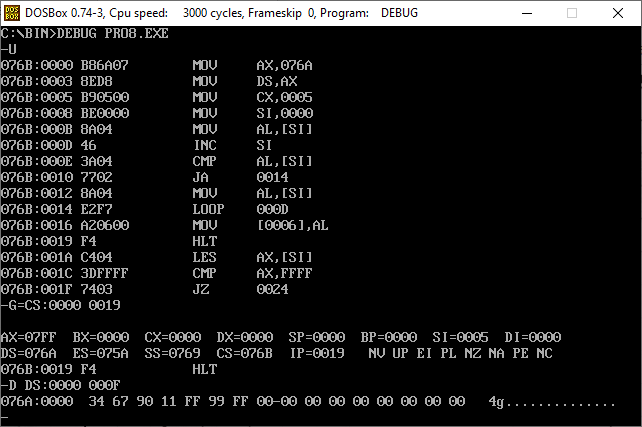
**Experiment Number-5 Smallest Number in a Given set of Numbers**

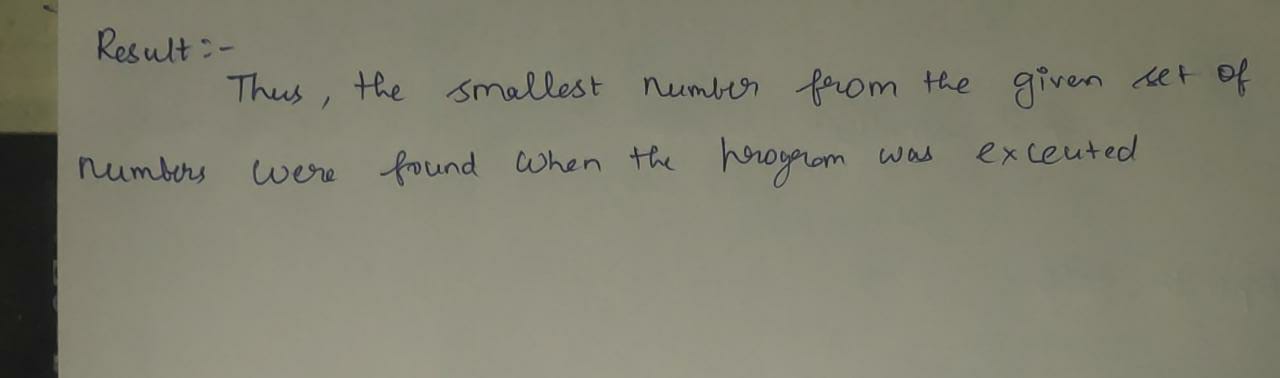
****

**Code:**

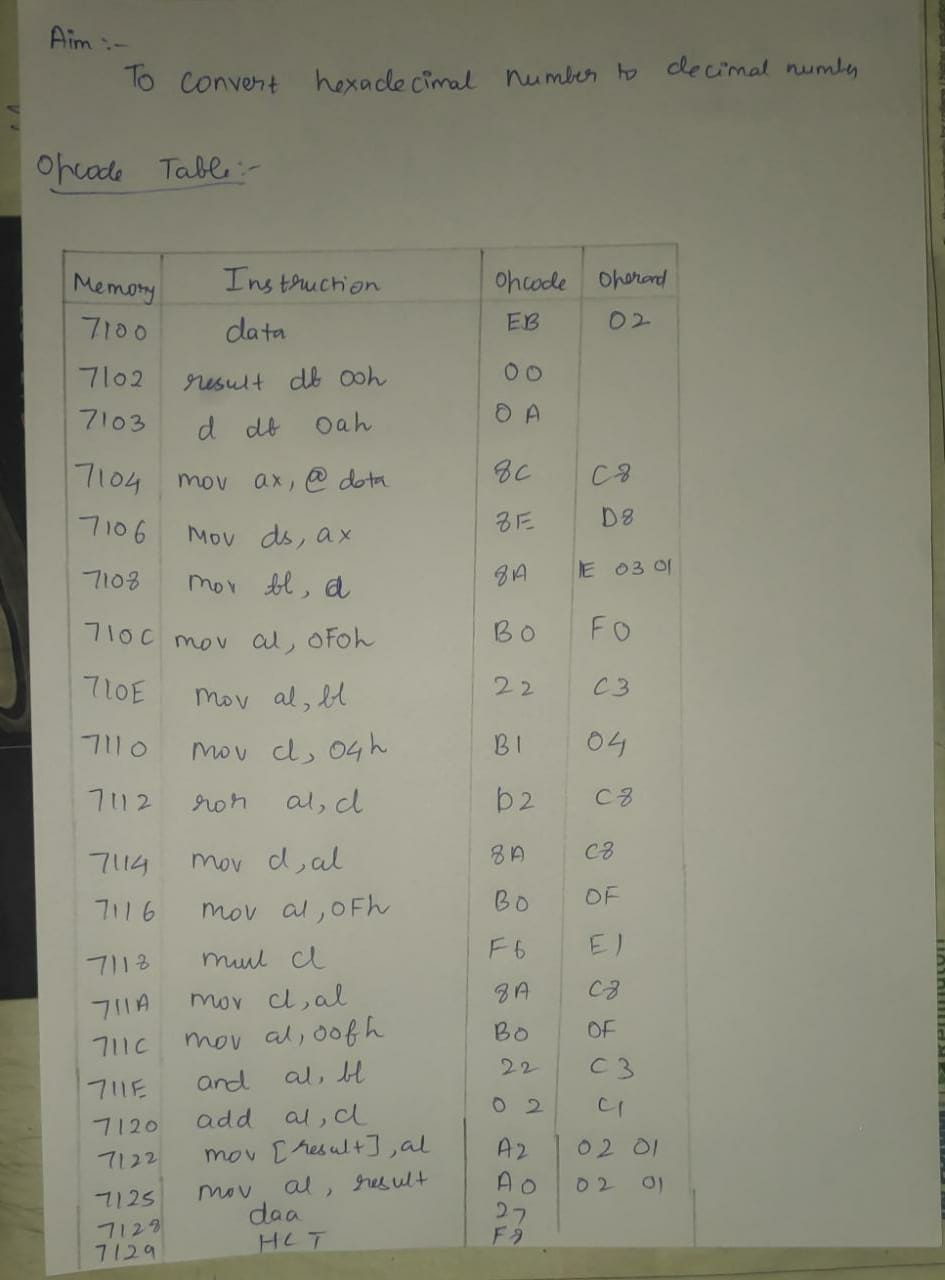


**Output:**



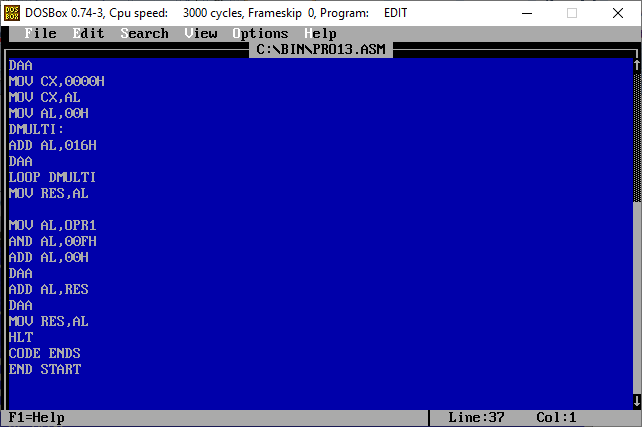
****

**Experiment Number-6 Convert Hexadecimal to Decimal Number**

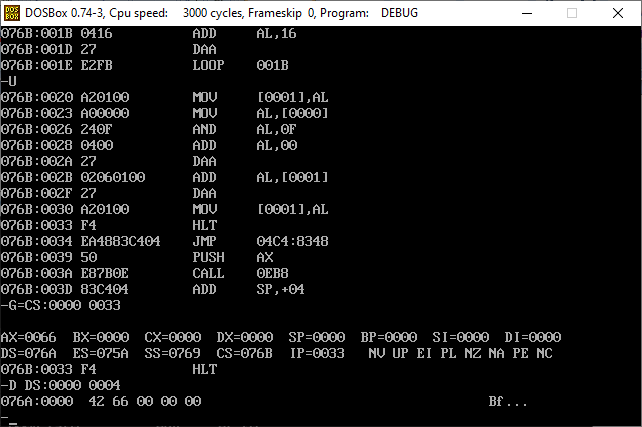
****

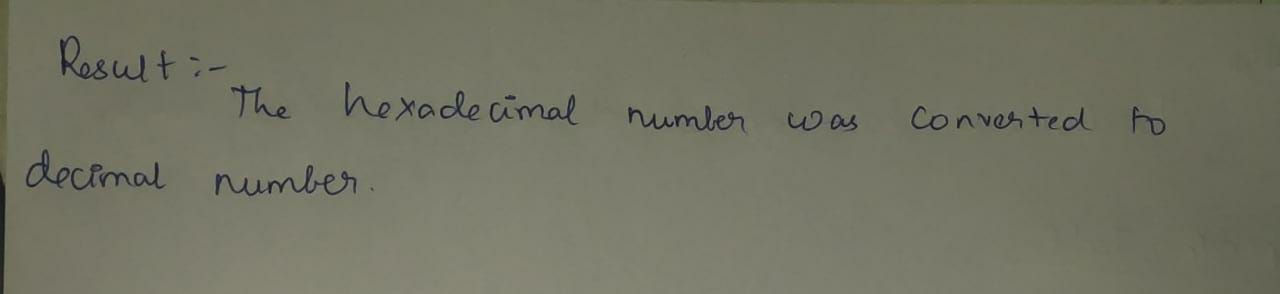
**CODE:**

****

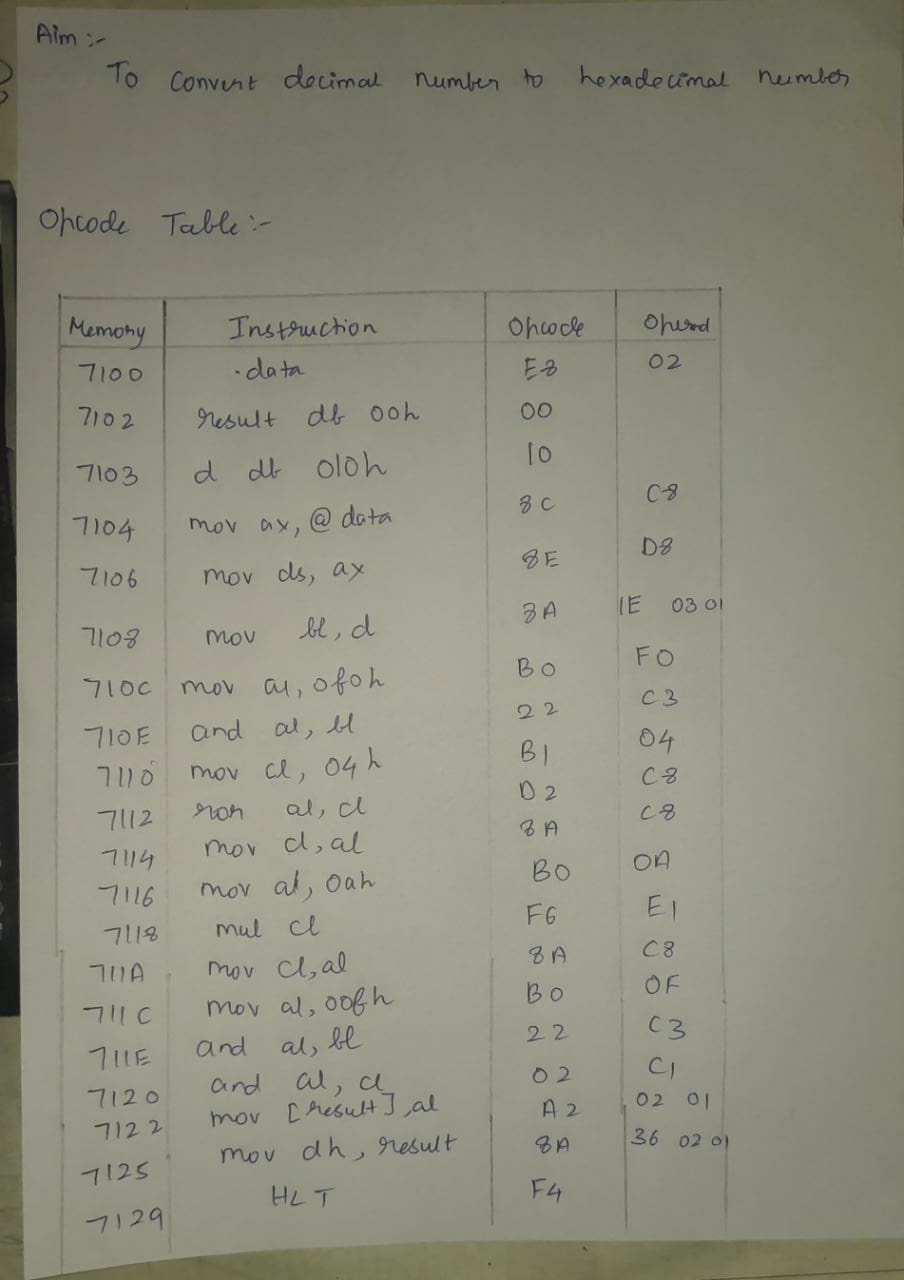
****

**OUTPUT:**

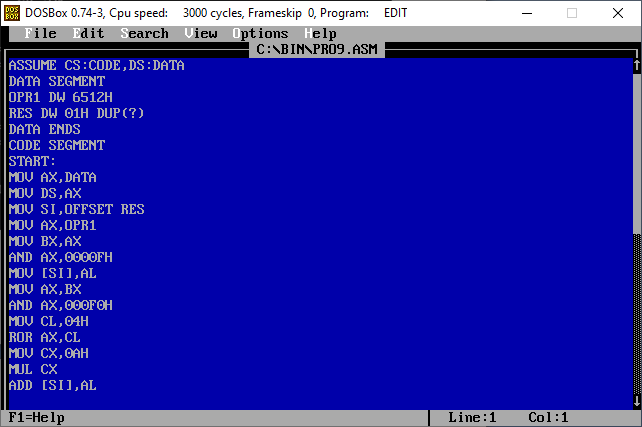
****

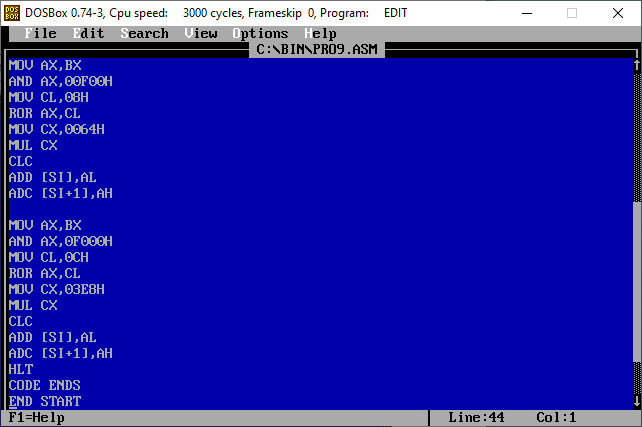
****

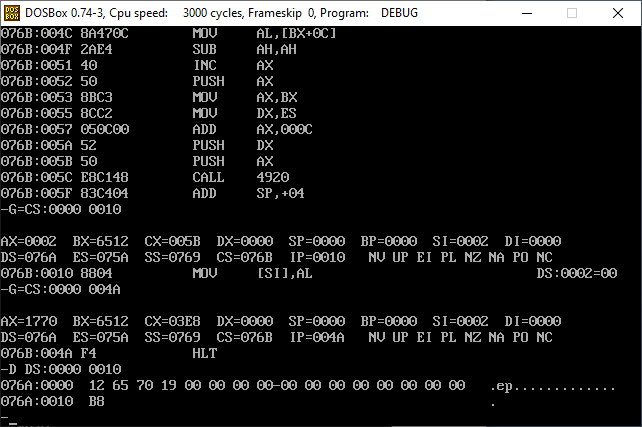
**Experiment Number-7 Convert Decimal to Hexadecimal Number**

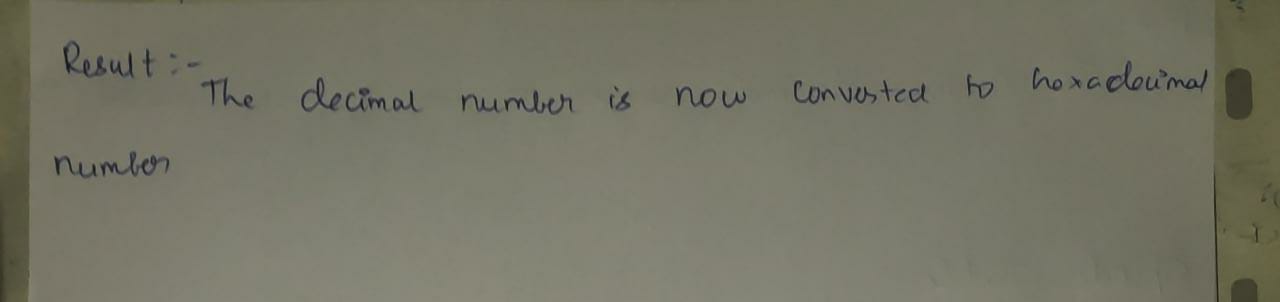
****

**CODE:**

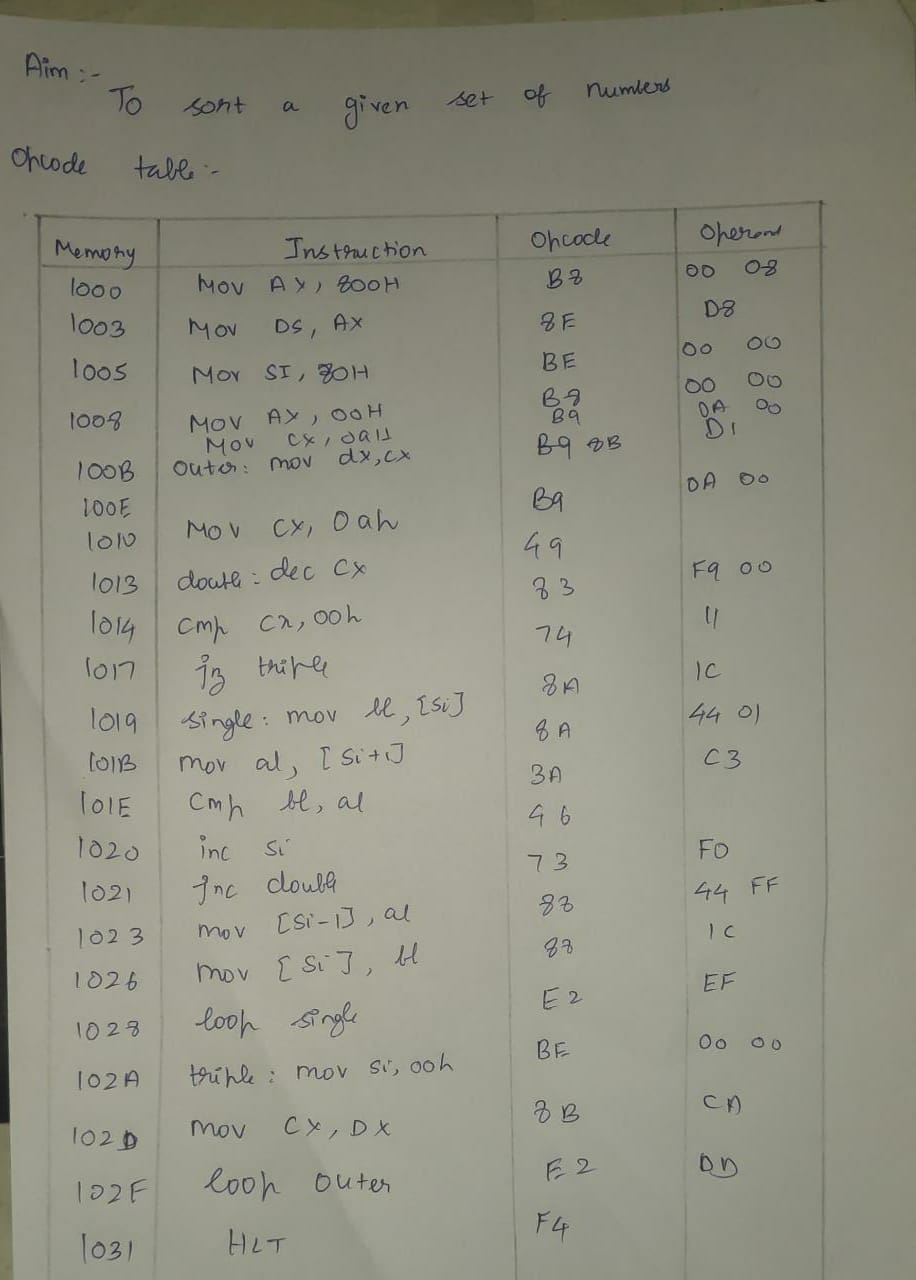
****

****

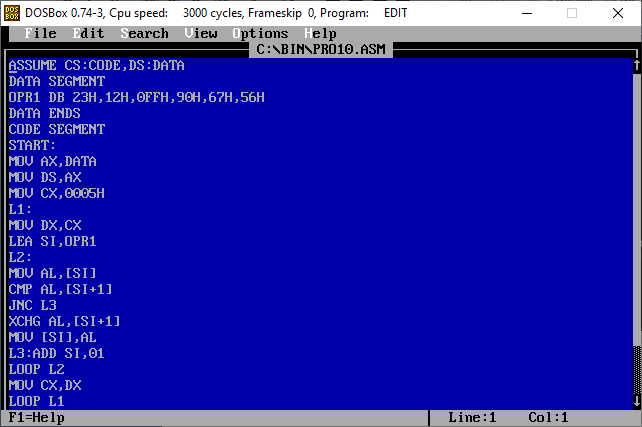
**OUTPUT:  
**

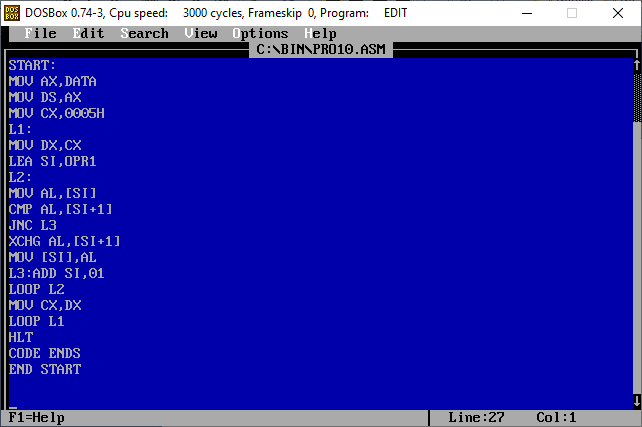
****

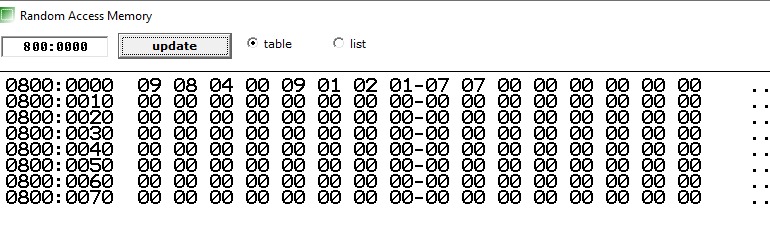
**Experiment Number-8 Sort the given set of Numbers**

****

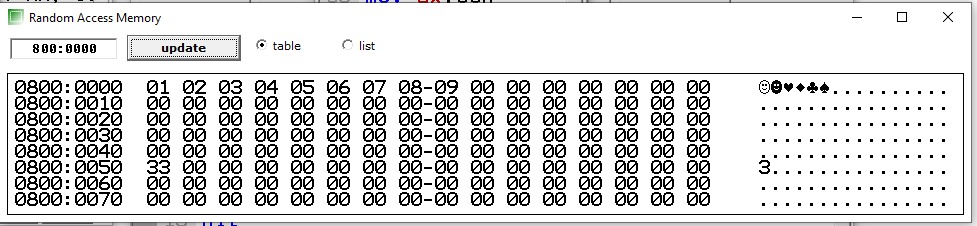
**CODE:**

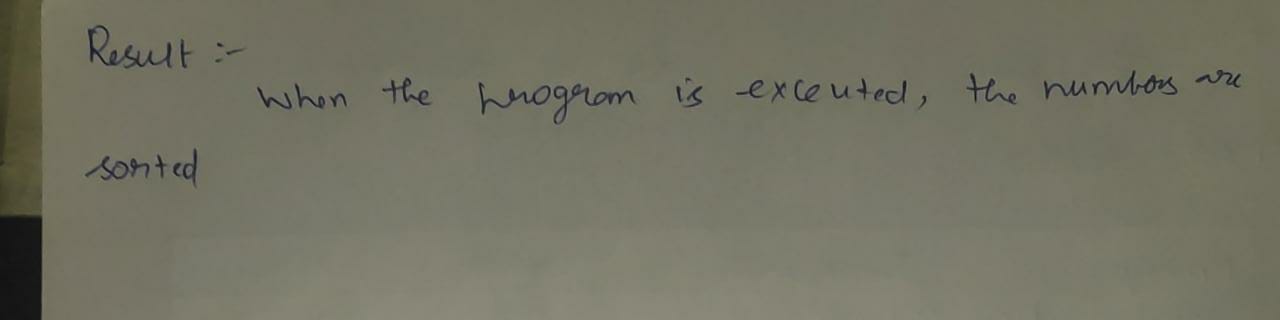
****

****

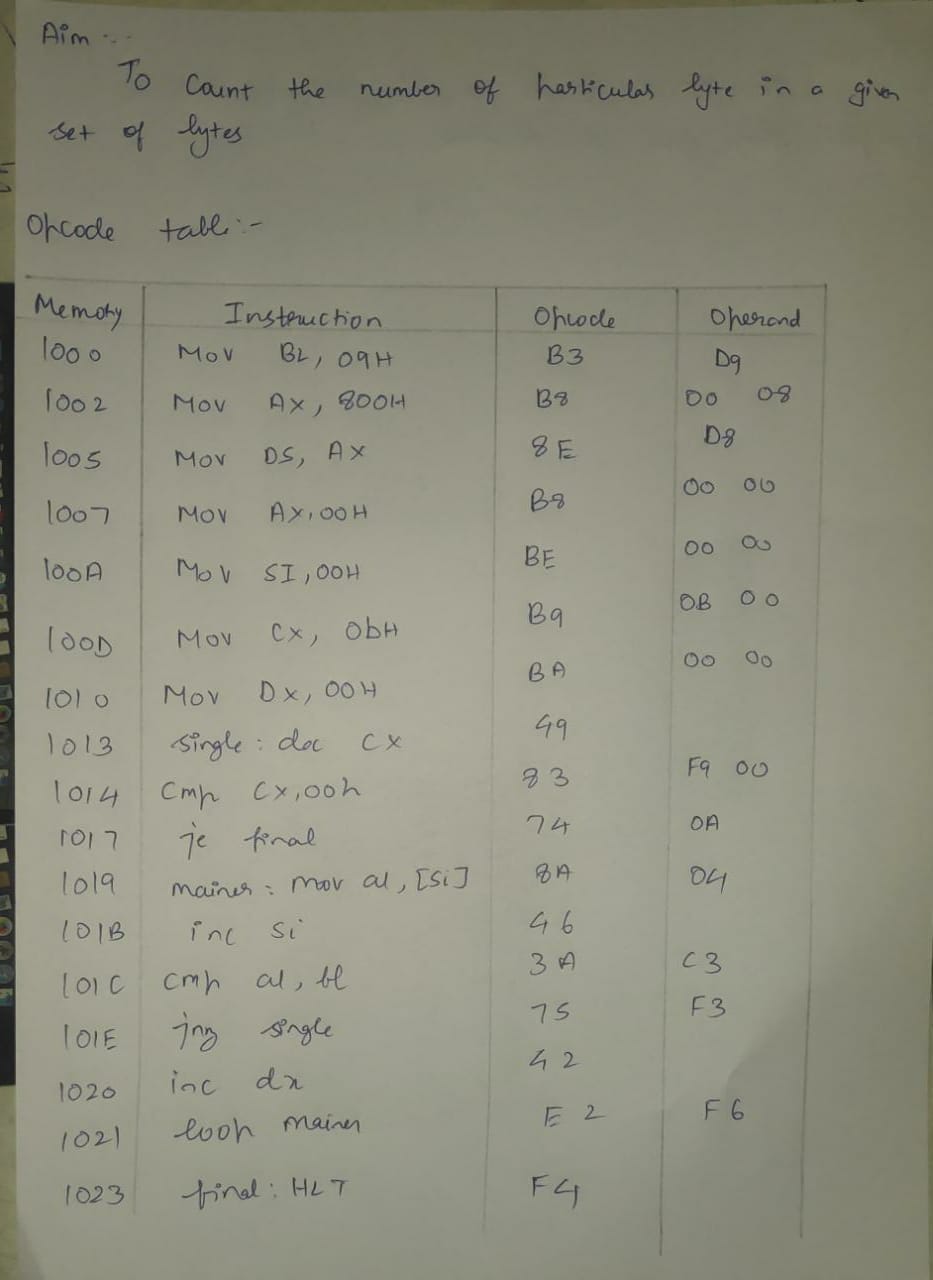
**Input:  
**

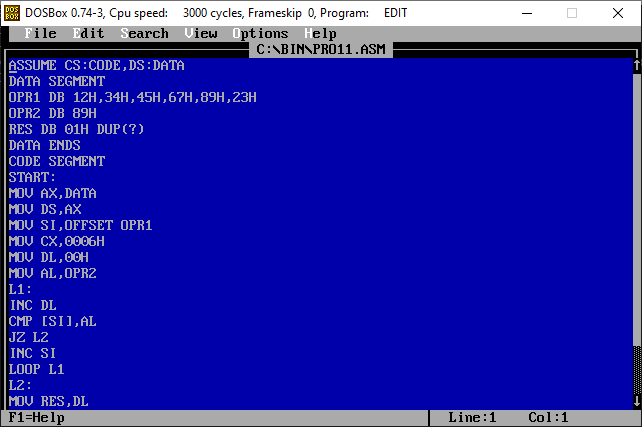
**Output:**

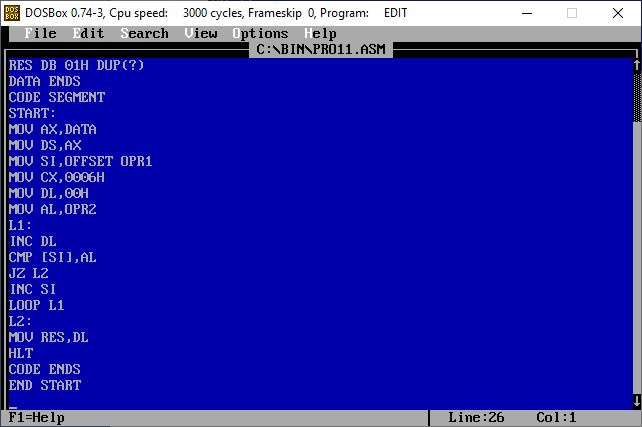
****

****

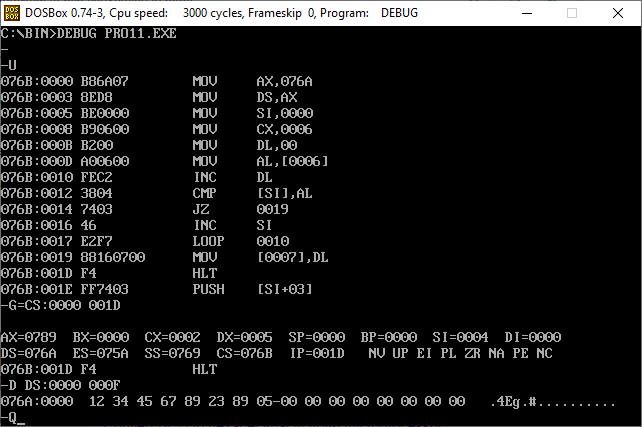
**Experiment Number-9 Count the number of particular byte in given set of byte**

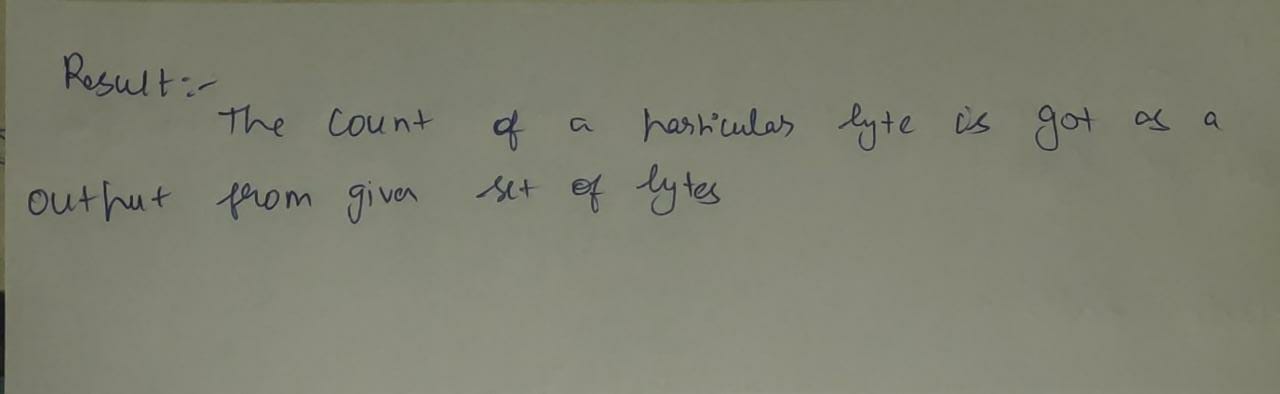
****

**CODE:  
**

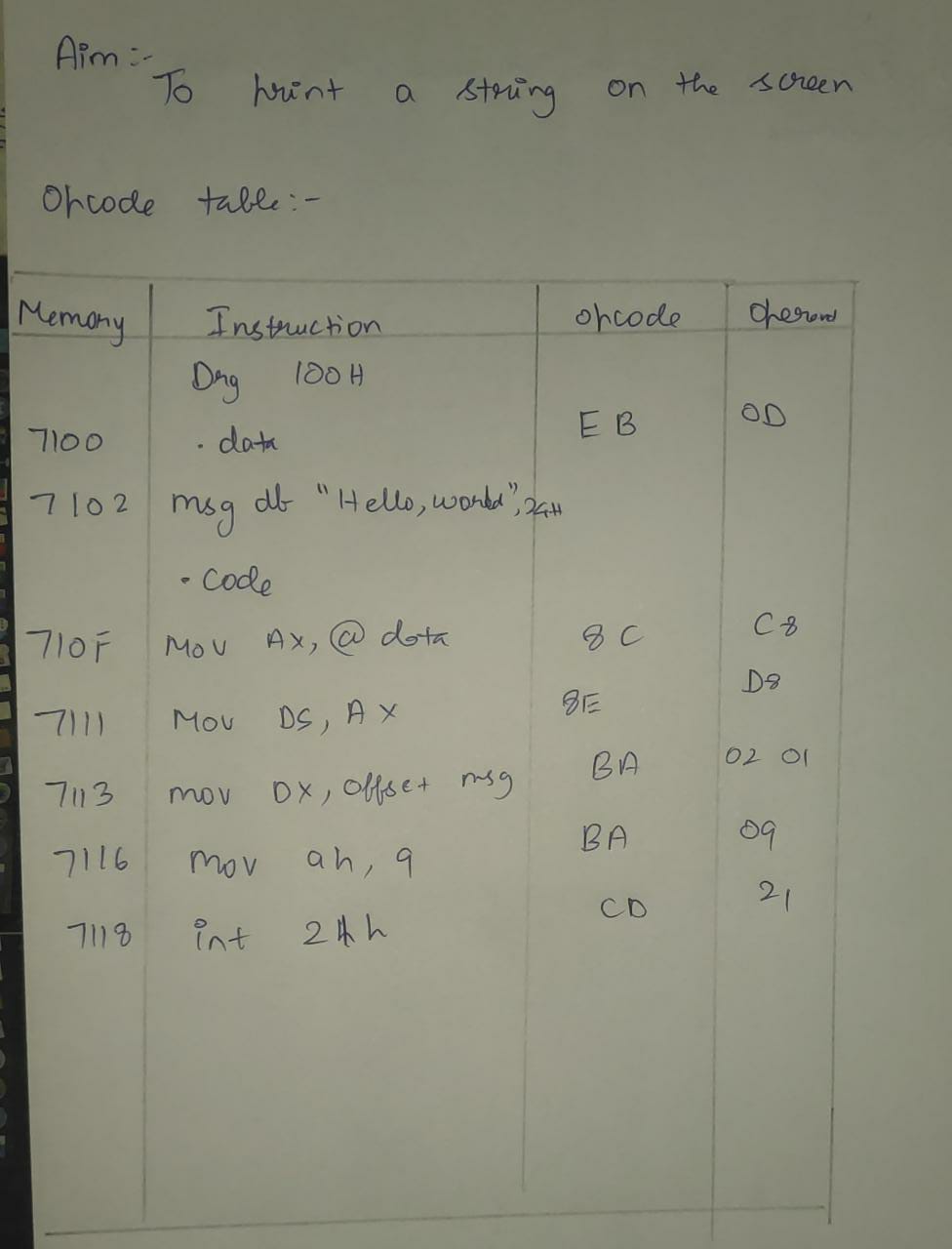
****

**OUTPUT:**

****

****

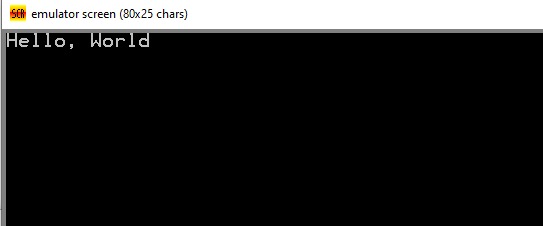
**Experiment Number-10 Print a String on the screen**

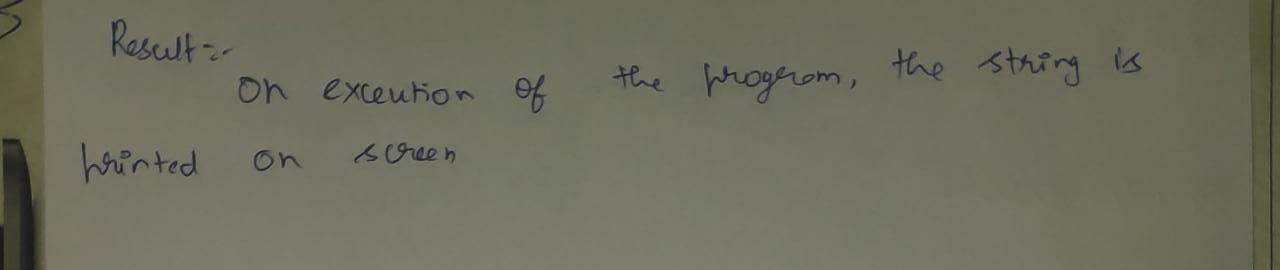
****

**Code:**



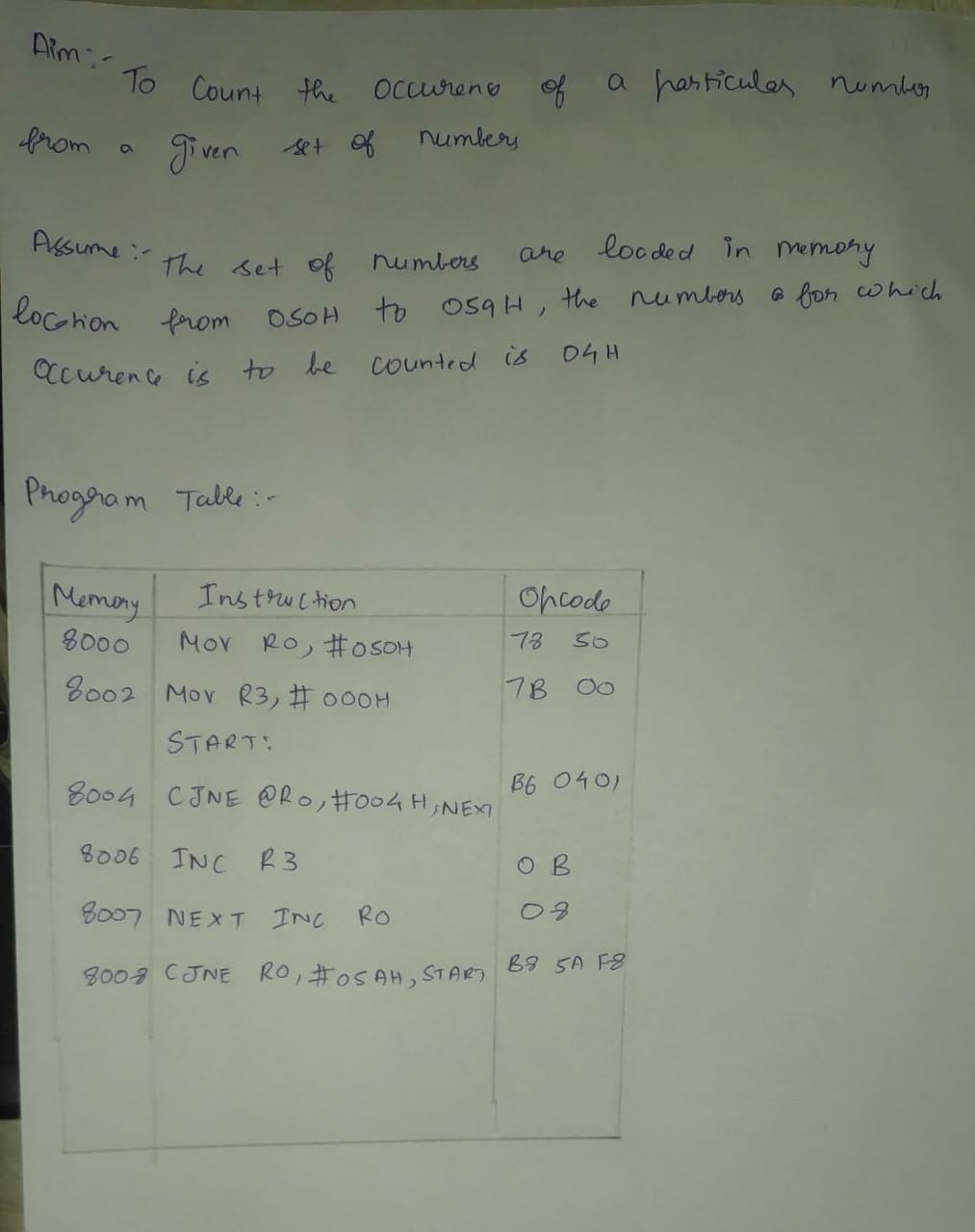
**Output:**

****

****

**ASSEMBLY LANGUAGE PROGRAMMING WITH 8051**

**Experiment Number-1 Count the Occurance of a Particular Number in a given set of number**

****

**Code:**

MOV 10H, #01H

MOV 11H, #02H

MOV 12H, #03H

MOV 13H, #01H

MOV R0, #10H

MOV R1, #04H

START: NOP

CJNE @R0, #01H, LOP

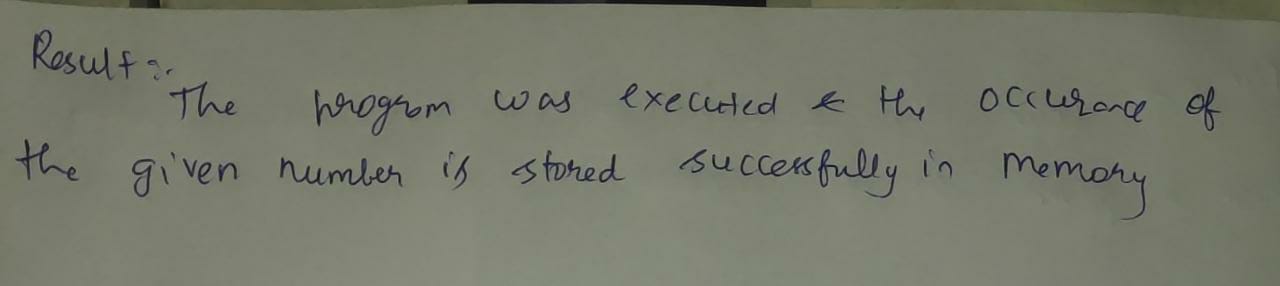
INC R0

INC R2

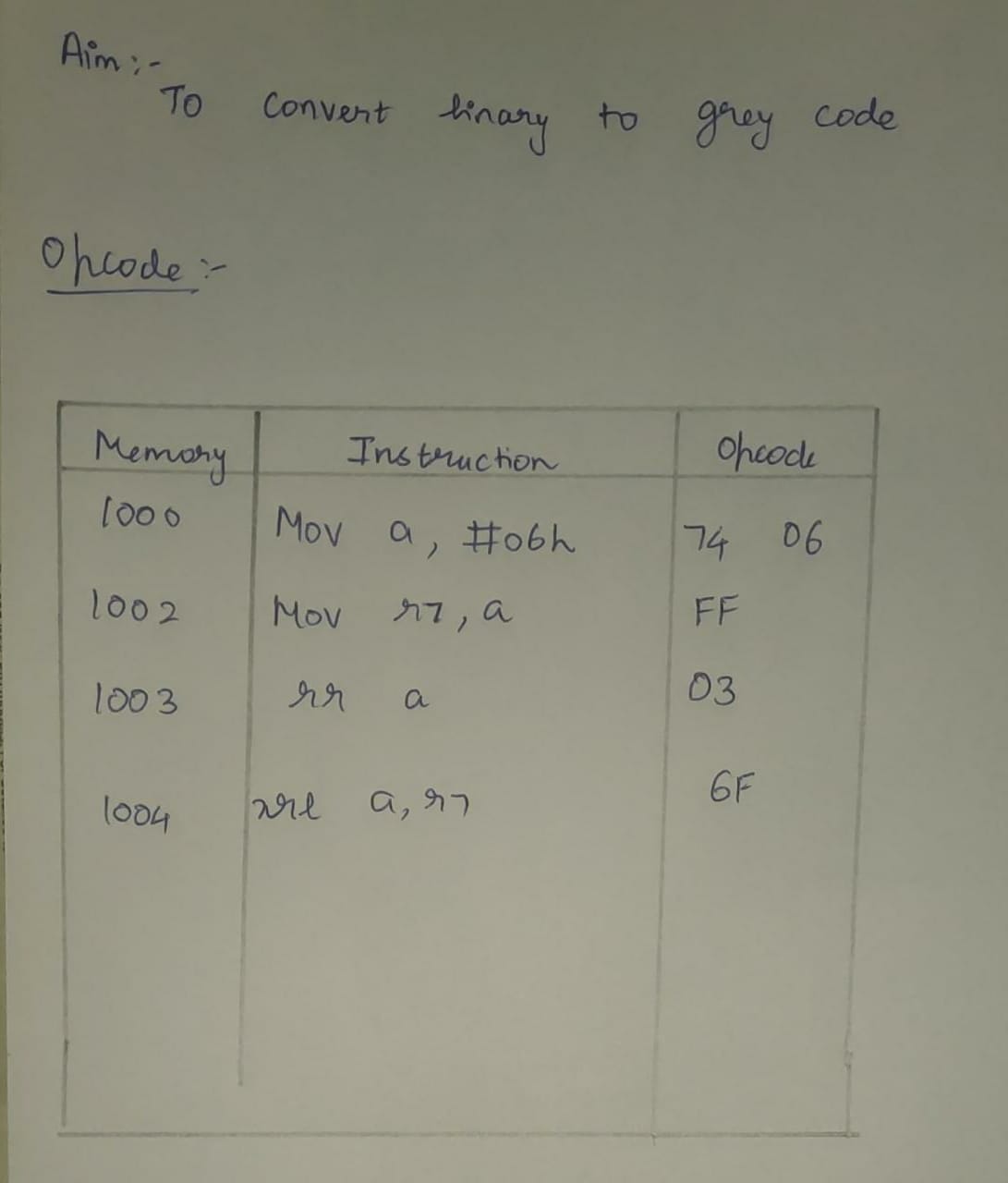
LOP: NOP

INC R0

DJNZ R1, START

****

**Experiment Number-2** **Convert Binary to Grey Code**



Code:

MOV 10H, #00H

MOV 11H, #00H ;BINARY DATA IN 10,11,12,13

MOV 12H, #01H

MOV 13H, #01H

MOV R0, #10H

MOV R1, #10H

MOV R2, #03H

START: NOP

MOV A, @R0

INC R1

XRL A, @R1 ;DO XOR FOR BITS 10,11; 11,12; 12,13 AND STORE IN 11, 12, 13

MOV @R1, A

INC R0

DJNZ R2, START ;LOOP THRICE

**INPUT:**

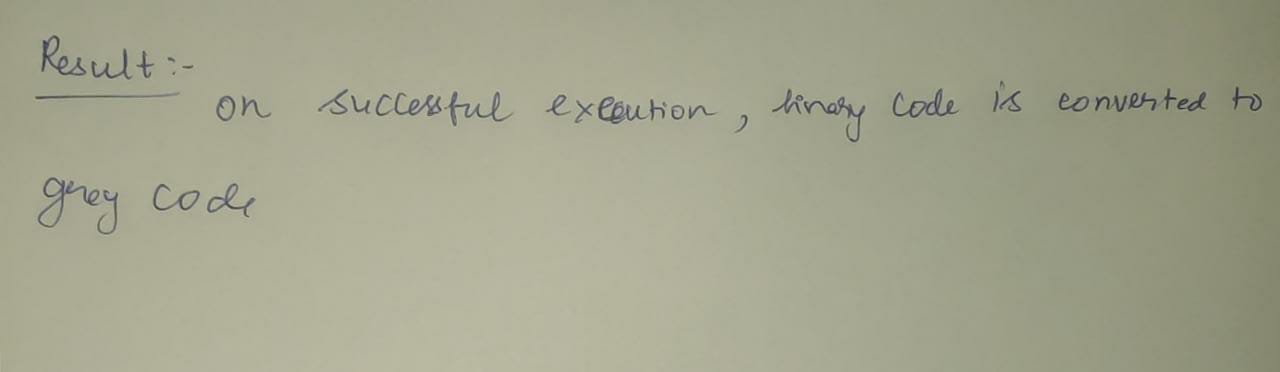
Table

Description automatically generated with low confidence

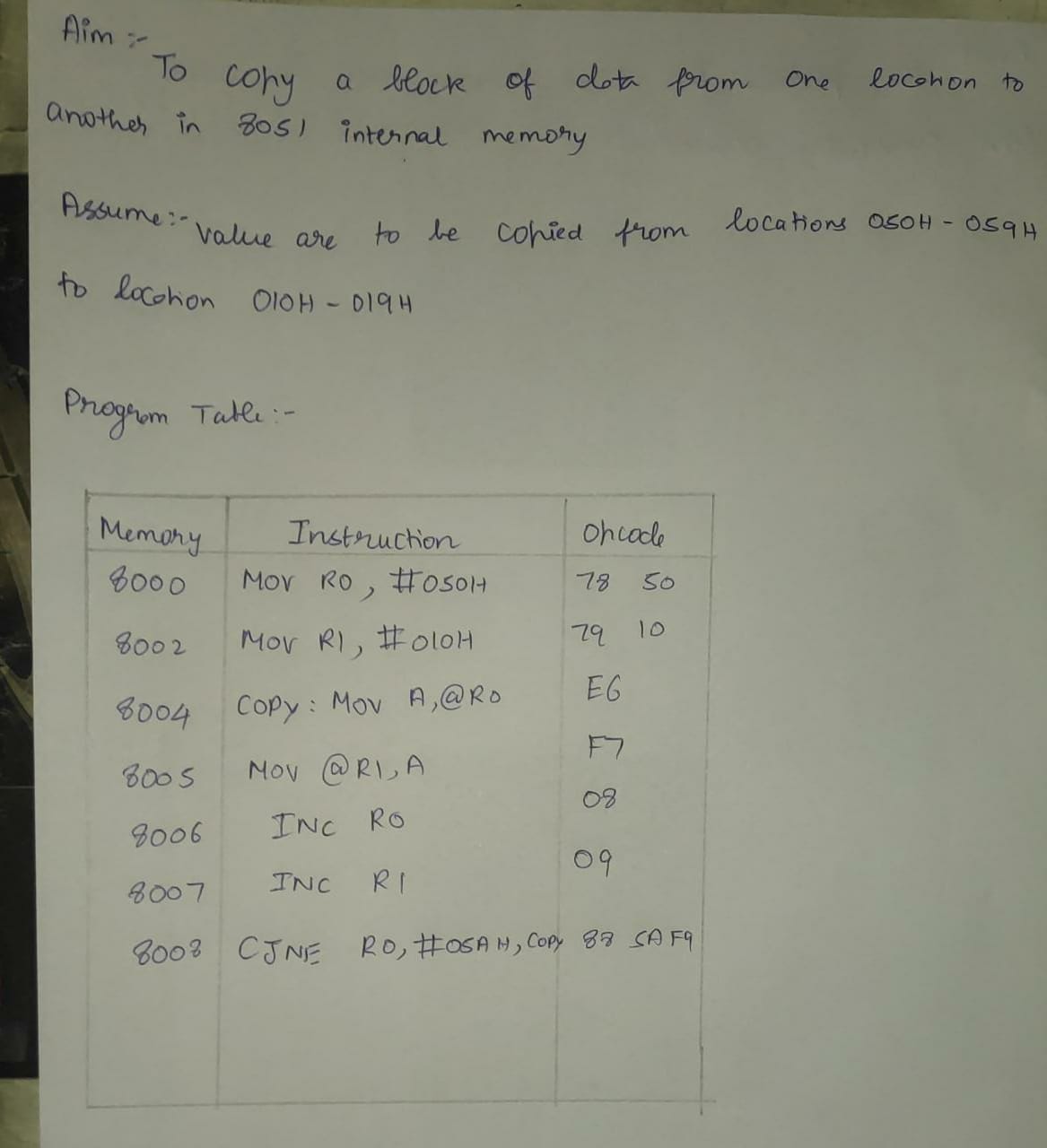
**OUTPUT:**

Table

Description automatically generated with low confidence



**Experiment Number-3 Copy a Block of data from one location to other**

****

**Code:**

MOV 10H, #01H

MOV 11H, #02H

MOV 12H, #03H

MOV 13H, #04H

MOV R2, #04H

MOV R1, #10H

MOV R0, #20H

LOOP: NOP

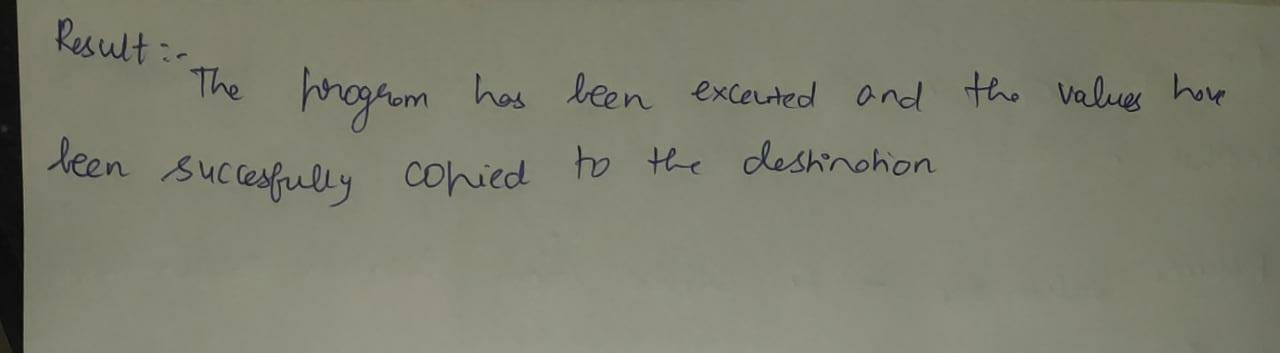
MOV A, @R1

MOV @R0, A

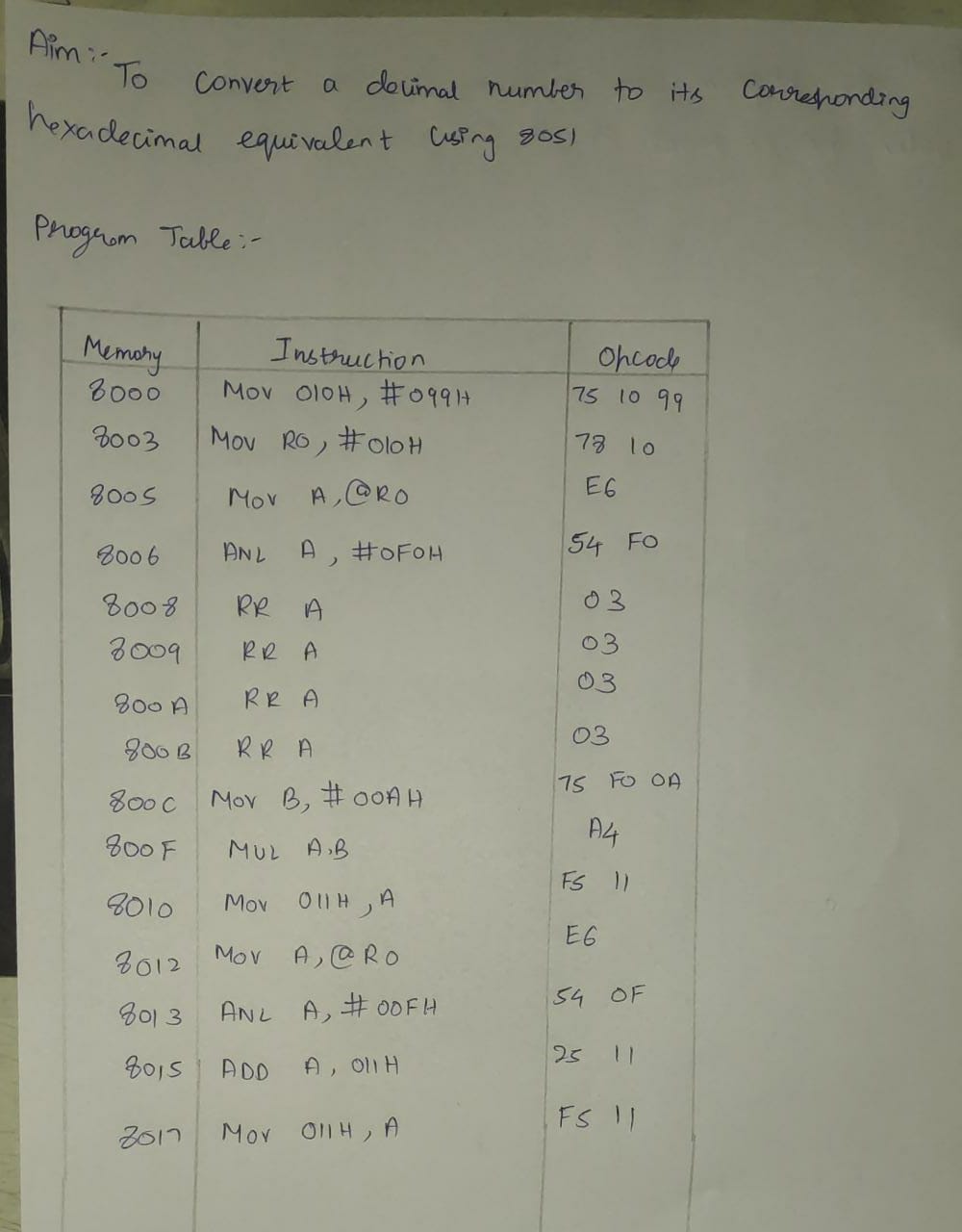
INC R1

INC R0

DJNZ R2, LOOP

****

**Experiment Number-4 Convert Decimal to Hexadecimal Number**

****

**Code:**

; here 99 is to be converted to hex

MOV 010H, #099H

; procedure starts

; store memory location in R0

MOV R0, #010H

; move the content to A

MOV A, @R0

; extract the 1st digit

ANL A, #0F0H

; rotate 4 times

RR A

RR A

RR A

RR A

; multiply with #A

MOV B, #00AH

MUL AB

MOV 011H, A

; extract next digit

MOV A, @R0

ANL A, #00FH

; no further processing needed

; add both

ADD A, 011H

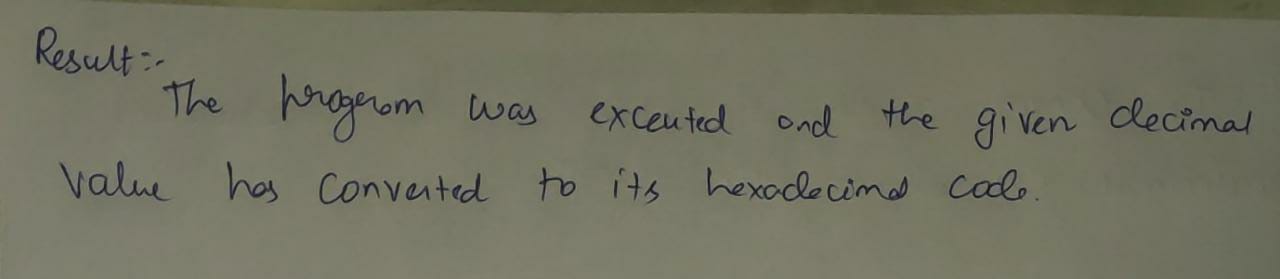
; store the result

MOV 011H, A

**Output:**

Graphical user interface, application

Description automatically generated

****

**Experiment Number-5 Convert Hexadecimal to Decimal Number**

****

**Code:**; store desired hex value in mem

MOV 010H, #02FH

; store address in R0

MOV R0, #010H

; extract 1st digit

MOV A, @R0

ANL A, #0F0H

DA A ; no need

; rotate 4 times

RR A

RR A

RR A

RR A

; store it in a mem location

MOV 011H, A

; multiply with decimal 16

MOV A, #000H

MULTIPLY: ADD A, #016H

DA A

DEC 011H

CJNE 011H, #000H, MULTIPLY

; store in a mem location

MOV 012H, A

; extract next digit

MOV A, @R0

ANL A, #00FH

DA A

; add with previous result

ADD A, 012H

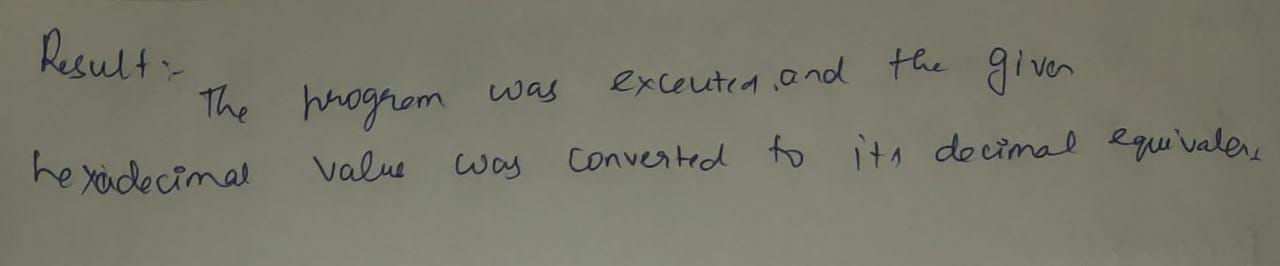
DA A

; store it

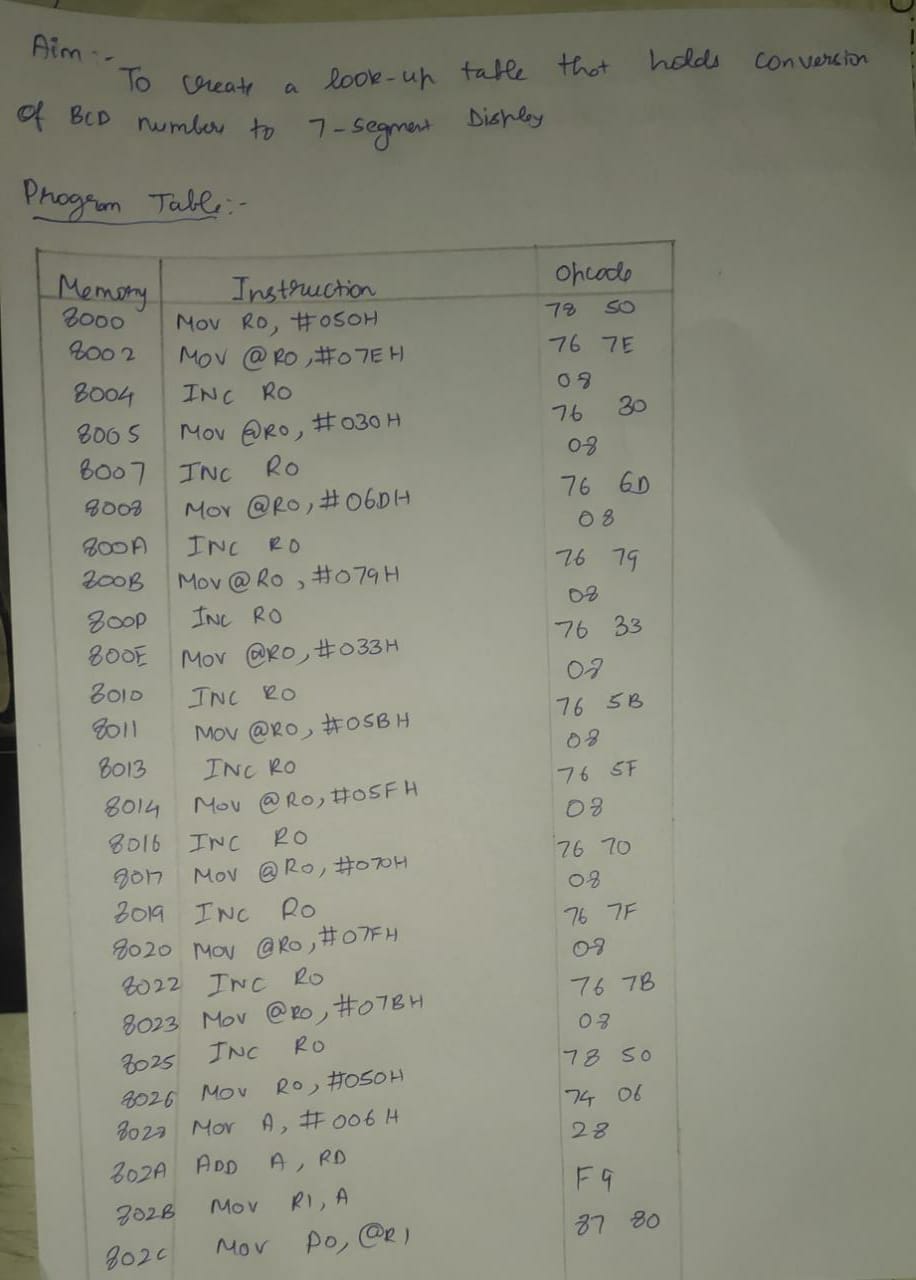
MOV 013H, A

**Output:**Graphical user interface, application

Description automatically generated

****

**Experiment Number-6 Lookup Table-7Segment**

****

**Code:**

; 7 segment display

; note: indexed addressing

; is possible only

; with DPTR or PC

; let start location = 050H

MOV R0, #050H

MOV @R0, #07EH ; code for 0

INC R0

MOV @R0, #030H ; code for 1

INC R0

MOV @R0, #06DH ; code for 2

INC R0

MOV @R0, #079H ; code for 3

INC R0

MOV @R0, #033H ; code for 4

INC R0

MOV @R0, #05BH ; code for 5

INC R0

MOV @R0, #05FH ; code for 6

INC R0

MOV @R0, #070H ; code for 7

INC R0

MOV @R0, #07FH ; code for 8

INC R0

MOV @R0, #07BH ; code for 9

; display

MOV R0, #050H ; starting address

MOV A, #006H ; input number

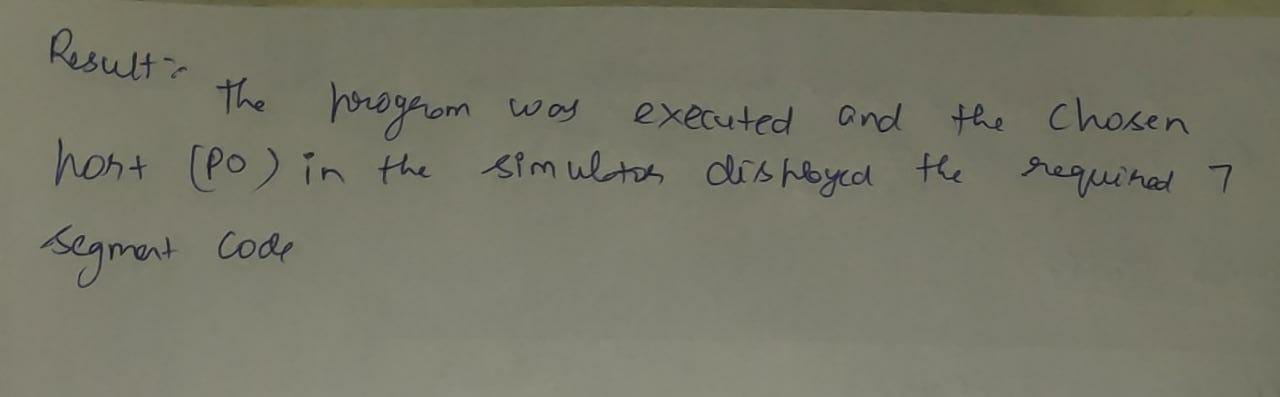
ADD A, R0 ; hold effectve address

MOV R1, A ; transfer it to R1

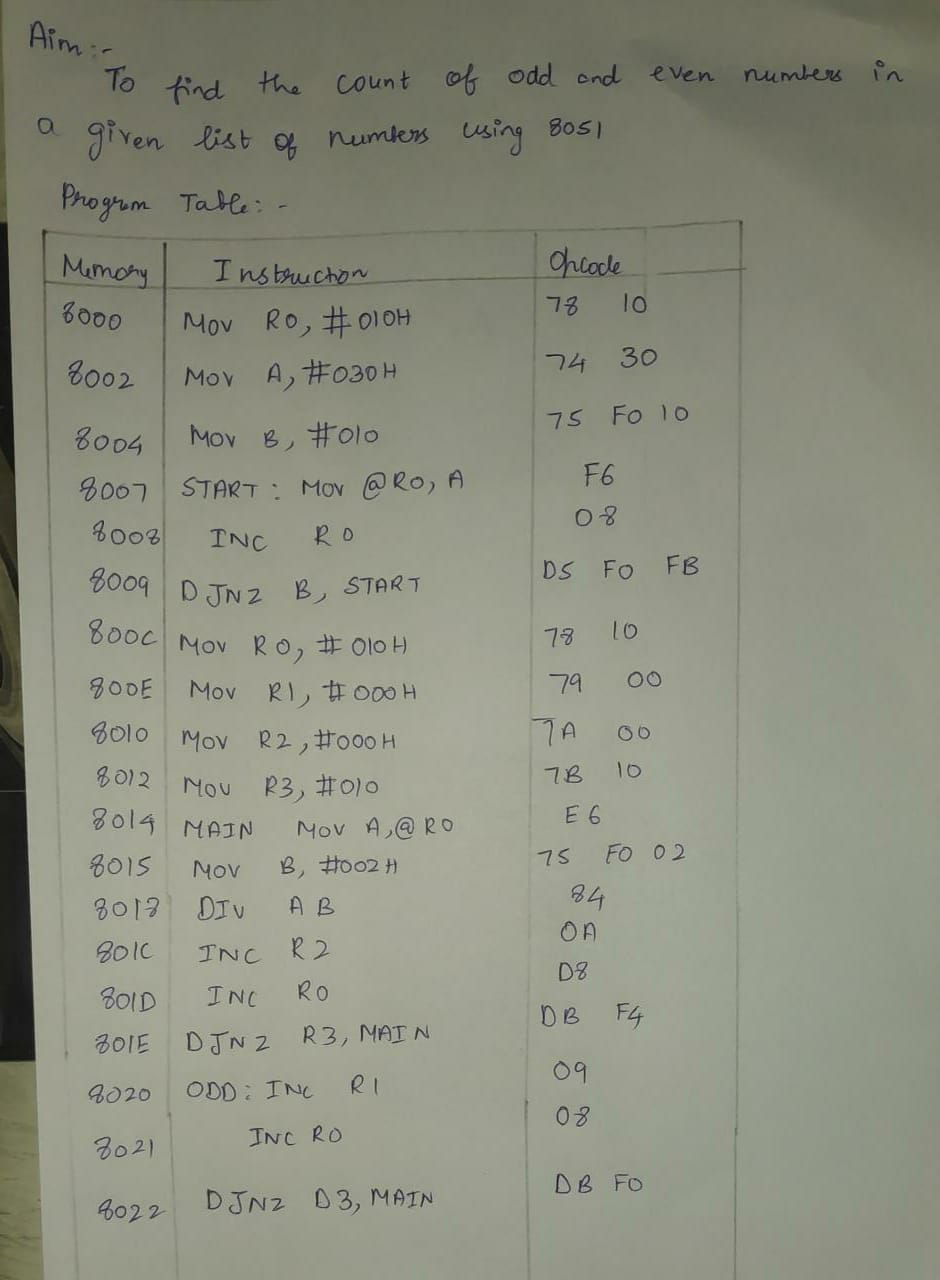
MOV P0, @R1 ; P1, P2 OR P3 CAN ALSO BE USED

**Output:**A picture containing text, stop, sign, clock

Description automatically generated

****

**Experiment Number-7 Count the number of odd and even numbers**

****

**Code:**

MOV 10H, #20H

MOV 11H, #21H

MOV 12H, #22H ;DATA

MOV 13H, #23H

MOV 14H, #24H

MOV R0, #10H

MOV R1, #05H

MOV R2, #00H

MOV R3, #00H

MOV B, #02H

START: NOP

MOV A, @R0

DIV AB

MOV A, B

CJNE A, #01h, EVEN

INC R3 ;STORES ODD NUMBER COUNT

INC R0

MOV B, #02H

EVEN: NOP

INC R0

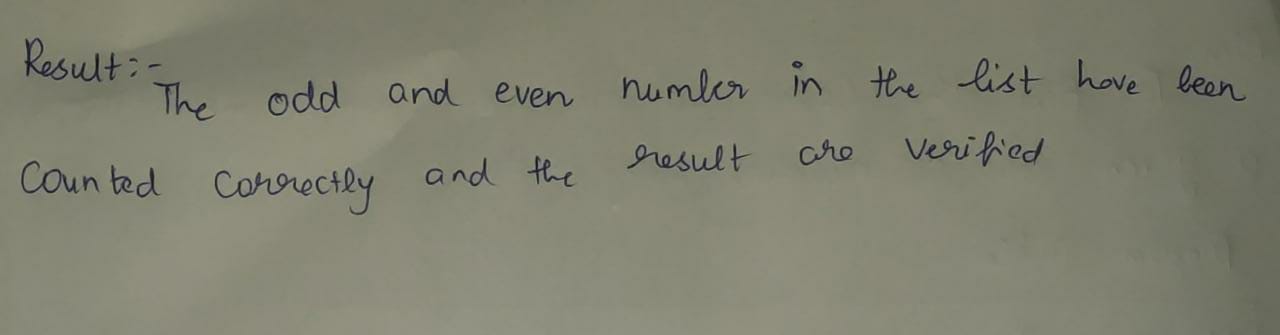
MOV B, #02H

DJNZ R1, START

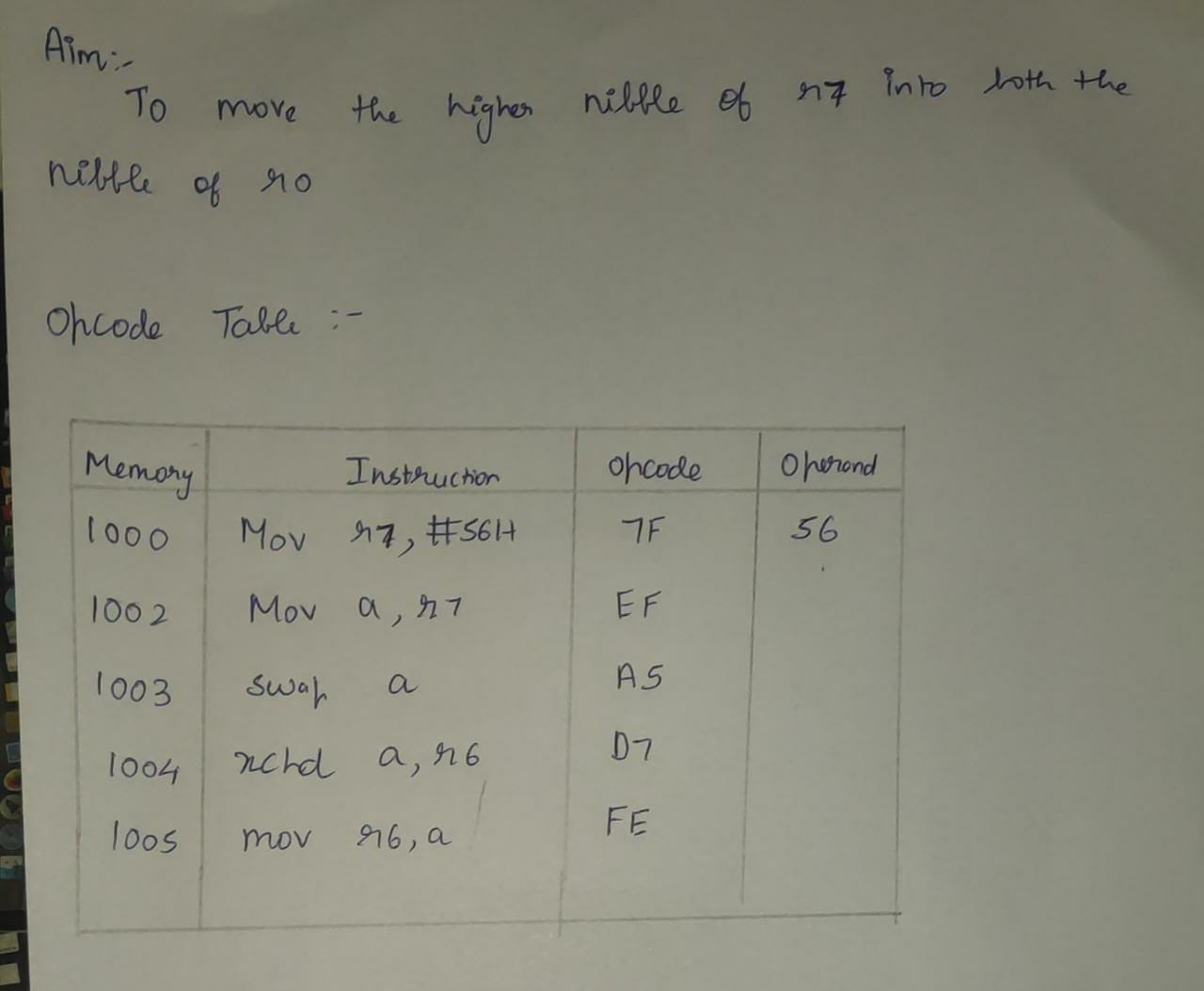
MOV A, #05H

SUBB A, R3 ;TO GET EVEN NUMBER COUNT

MOV R2, A ;STORES EVEN NUMBER COUNT

****

**Experiment Number-8 Move Higher nibble in to both the nibble of r0**

****

**Code:**; to store higher nibble of r7

; in both nibbles of r6

MOV R7, #0FEH

MOV A, R7

; extract the higher nibble

ANL A, #0F0H

; move it to r6

MOV R6, A

; rotate the content

; 4 times to make it

; as lower nibble

RR A

RR A

RR A

RR A

; or it with r6

ORL A, R6

MOV R6, A

**INPUT:**

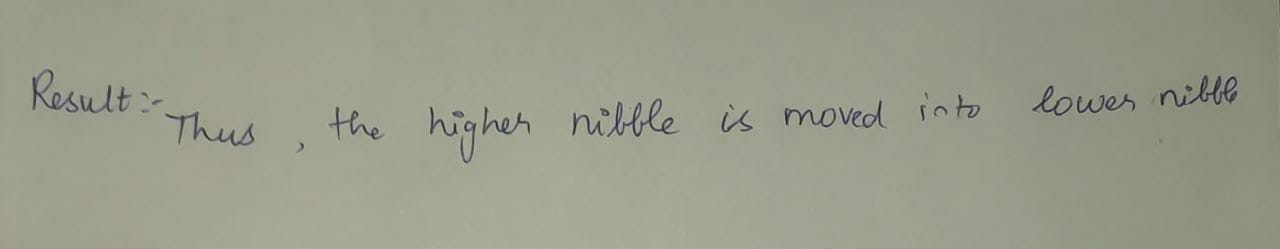
Graphical user interface, text, application

Description automatically generated

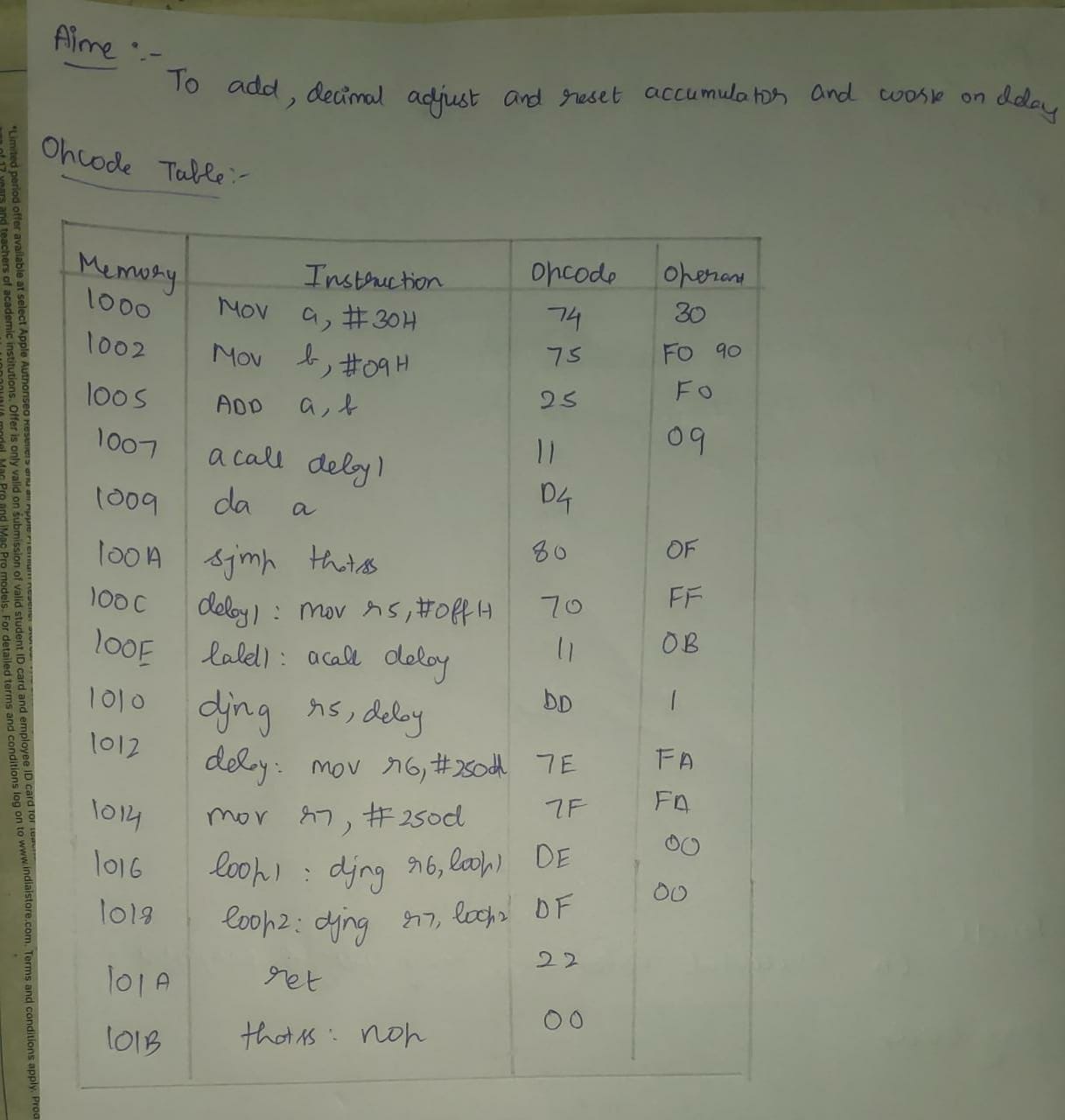
**OUTPUT:**

Graphical user interface, application

Description automatically generated with medium confidence

****

**Experiment Number-9 ADD and do reset accumulator with a delay**



Code:

MOV A, #30H

MOV B, #0AH

ADD A,B

DA A

ACALL DELAY1

DELAY1: MOV R5,#10D

LABEL: ACALL DELAY

DJNZ R5,LABEL

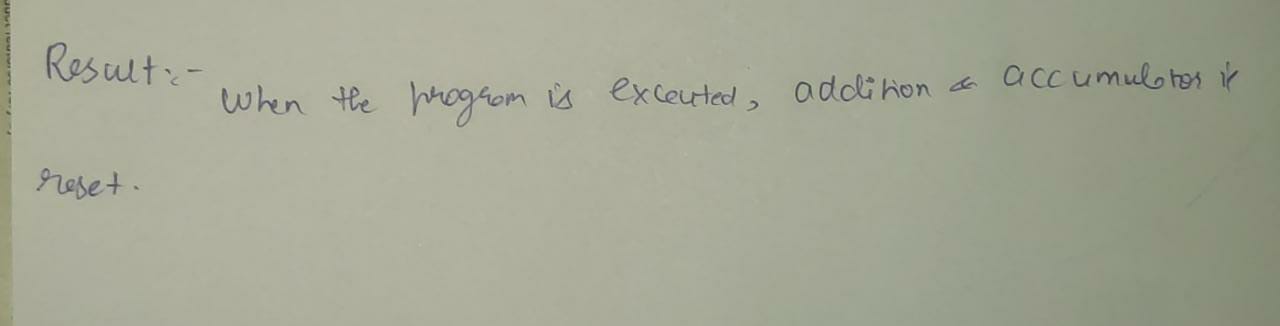
DELAY: MOV R6,#250D

MOV R7,#250D

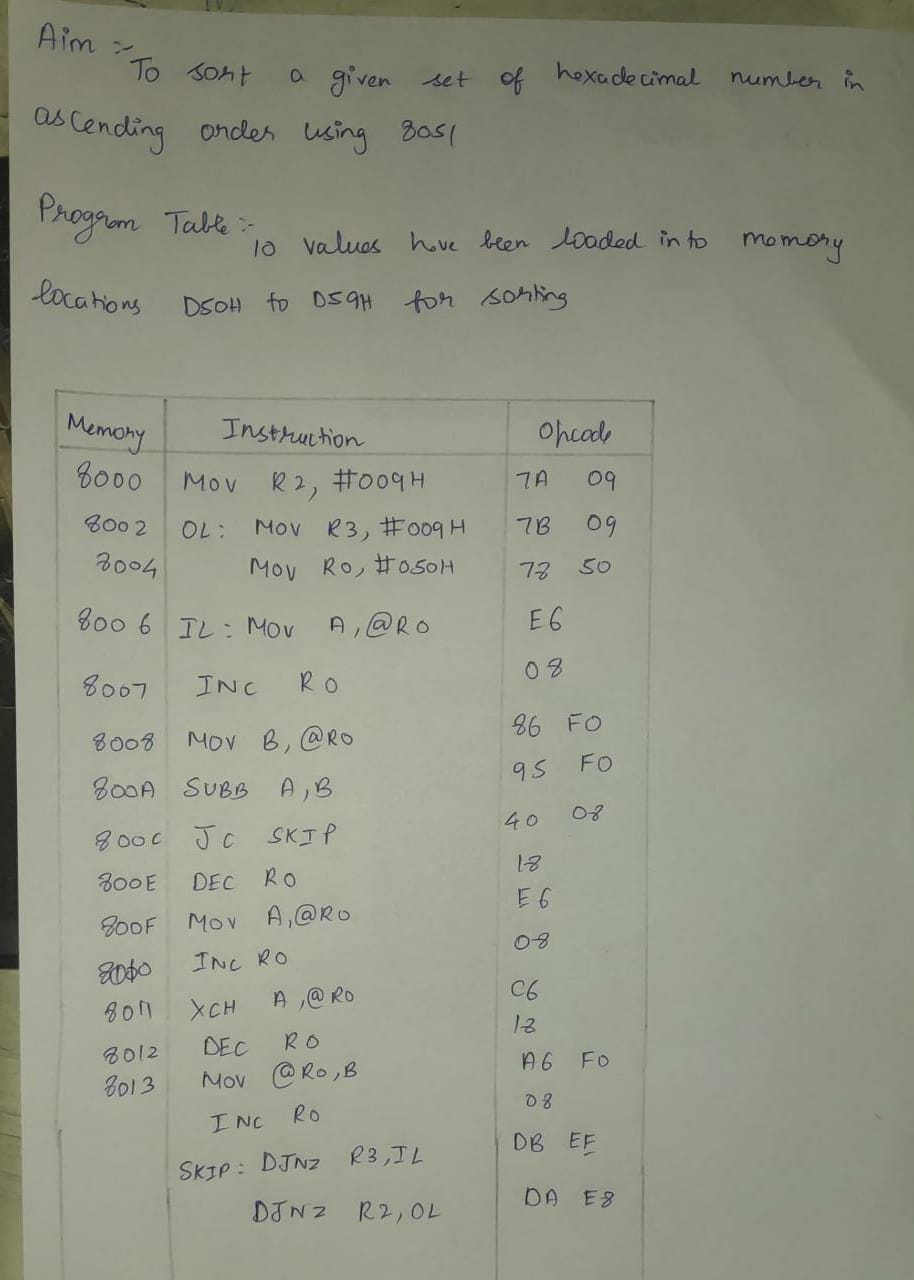
LOOP1: DJNZ R6,LOOP1

LOOP2: DJNZ R7,LOOP1

;MOV A, #00H

****

**Experiment Number-10 Sort Numbers in ascending order**

****

**Code:**

ORG 0000H  
MOV R7,#4  
loop1:MOV R0,#40H  
 MOV R6,#04  
loop:MOV A,@R0  
 INC R0  
 MOV 50H,@R0  
 CJNE A,50H,next  
 SJMP down  
next:JC down  
 MOV @R0,A  
 DEC R0  
 MOV @R0,50H  
down:DJNZ R6,loop  
 DJNZ R7,loop1  
END

**INPUT:**

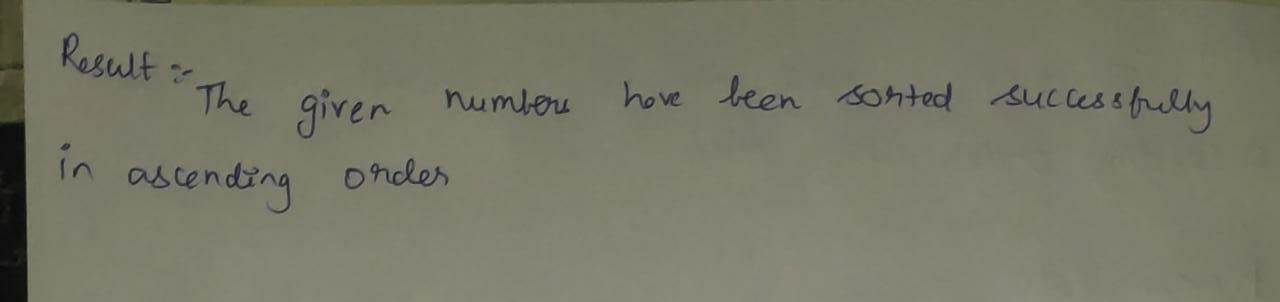
Table

Description automatically generated

**OUTPUT:**

Table

Description automatically generated

****