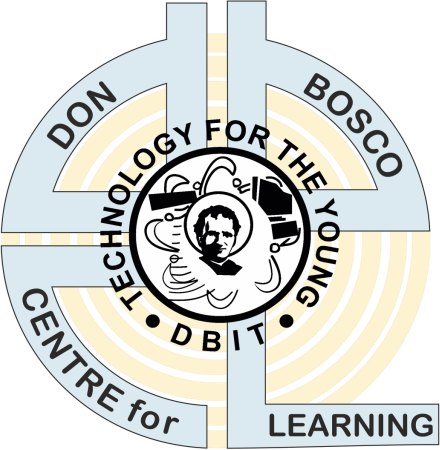
**Lab Journal**



Name : ANIKET VINAYAK PARATE

Roll No. 38 Seat No. 3446

Course/Lab: ITL403

Semester: IV

Academic Year: 2021-2022

**Subject In-charge:**

Department of Information Technology

Don Bosco Institute of Technology, Mumbai 400070.

INDEX PAGE

|  |  |  |  |
| --- | --- | --- | --- |
| EXPERIMENT NO. | TITLE | PAGE NO. | DATE |
| 1. | To study the truth table of Logic gates | 3 | 04.02.2022 |
| 2. | To study the truth table of Full adder | 7 | 04.02.2022 |
| 3. | To study MUX and DeMUX | 9 | 15.02.2022 |
| 4. | Study of PC mother board | 12 | 20.02.2022 |
| 5. | Arithmetic operation using  EMU8086 | 13 | 29.02.2022 |
| 6. | A program to calculate the number of zeros and ones in an 8 bit number | 17 | 23.03.2022 |
| 7. | A program to find the largest and smallest number out of 5  numbers | 19 | 23.03.2022 |
| 8. | To calculate a factorial of a number. | 21 | 04.04.2022 |
| 9. | To calculate whether a given string is a palindrome or not | 23 | 04.04.2022 |
| ASSIGNMENT NO. | | PAGE NO. | DATE |
| 1. Assignment      1. Assignment | | 25 | 28.03.2022 |
| 27 | 28.03.2022 |

Performance date : 04.02.2022

# EXPERIMENT NO. 1

Aim : To study the truth table of Logic gates

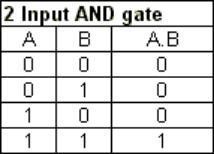
Tool used: Logisim

Theory:

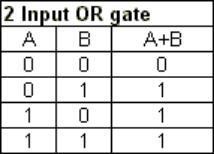
Write the truth table of the following gates: AND, OR,NOT, NAND,

NOR.

1. AND GATE:



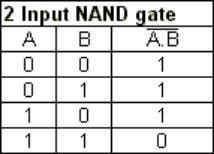
1. OR GATE :



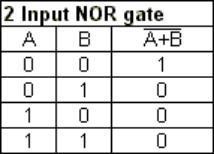
1. NOT GATE:



1. NAND GATE:

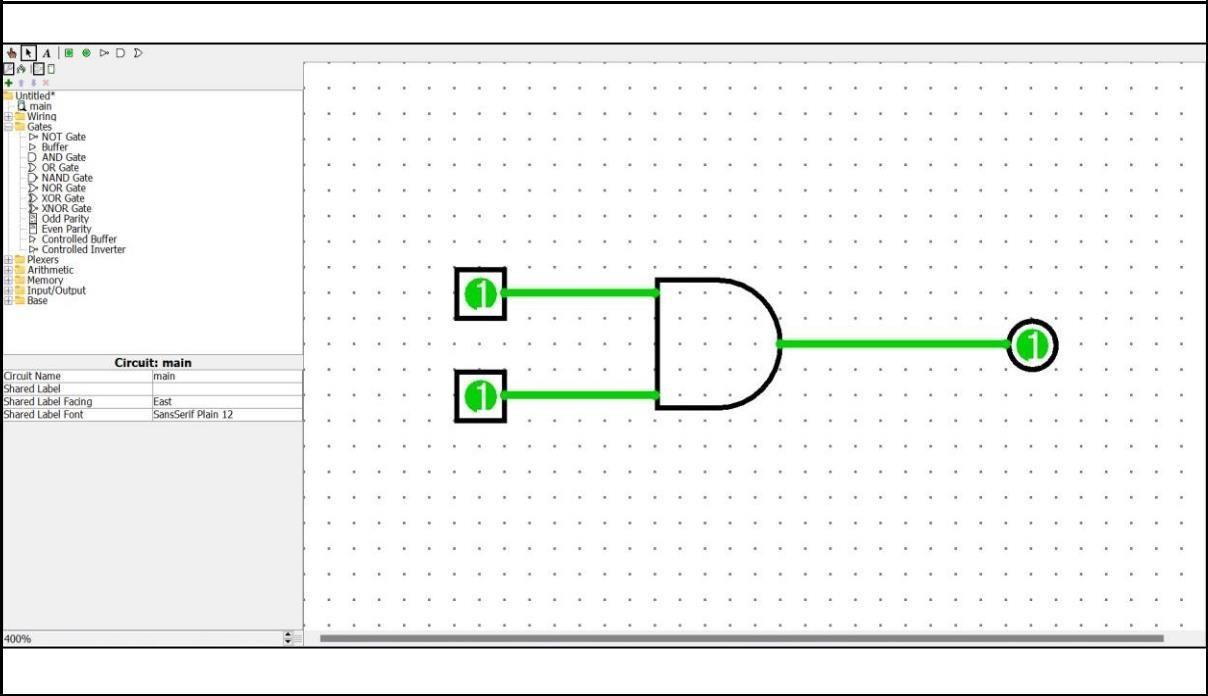


1. NOR GATE

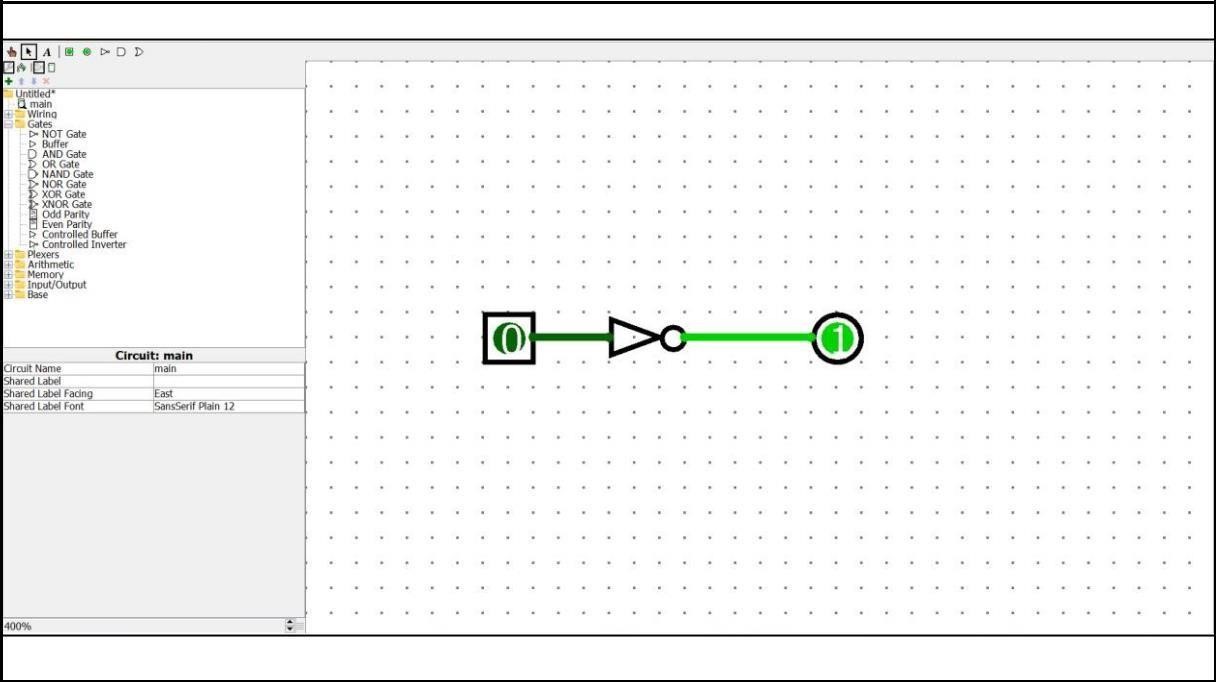


**OUTPUT:-**

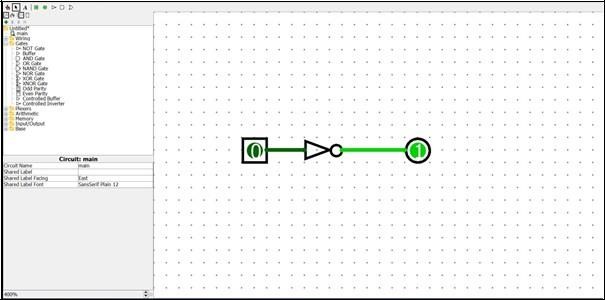
* 1. AND GATE:



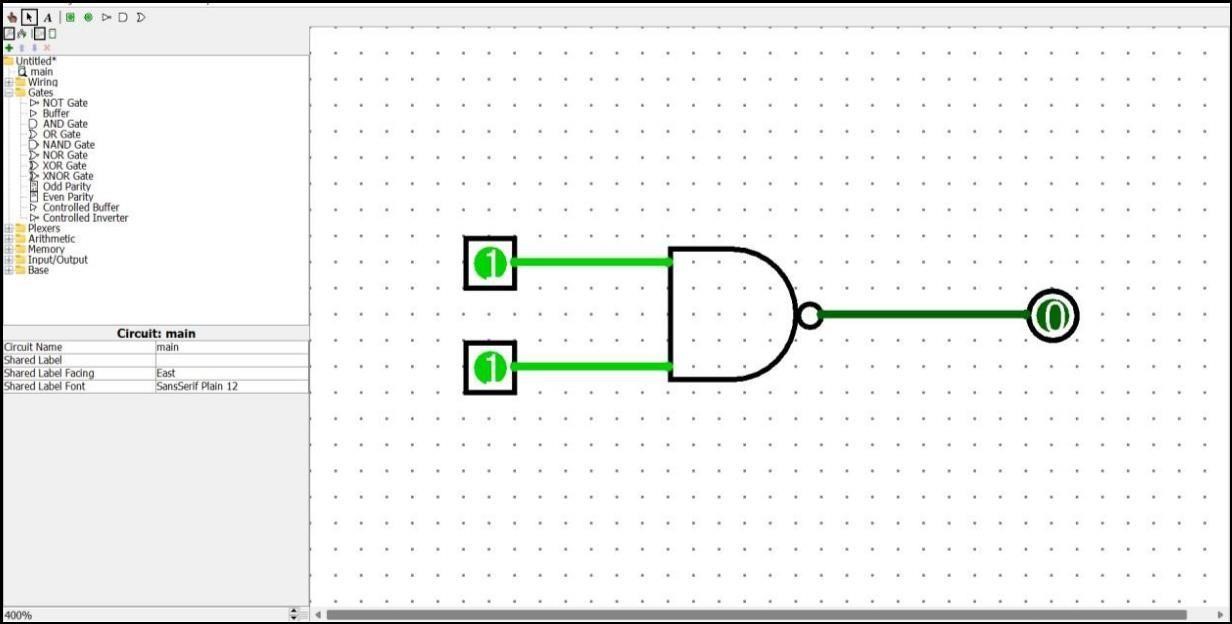
* 1. OR GATE:



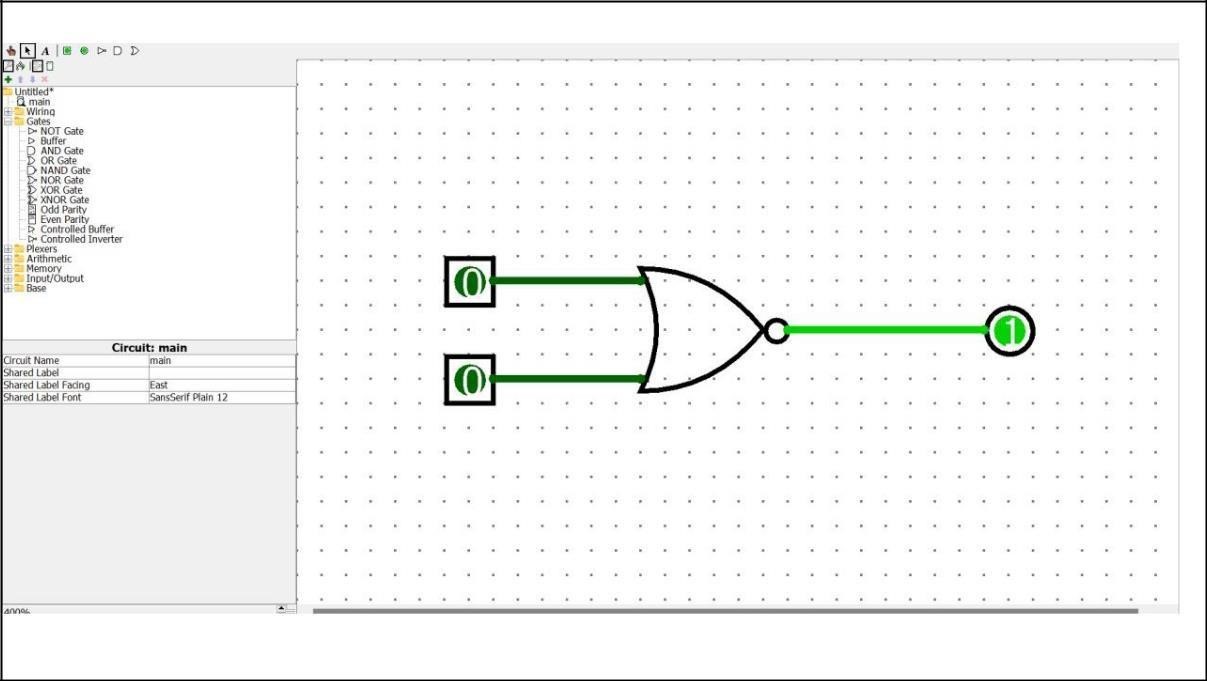
* 1. NOT GATE :



* 1. NAND GATE:



* 1. NOR GATE:



Performance date : 04.02.2022

# EXPERIMENT NO. 2

Aim : To study the truth table of Full adder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | SUM | C out |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Tool used: Logisim

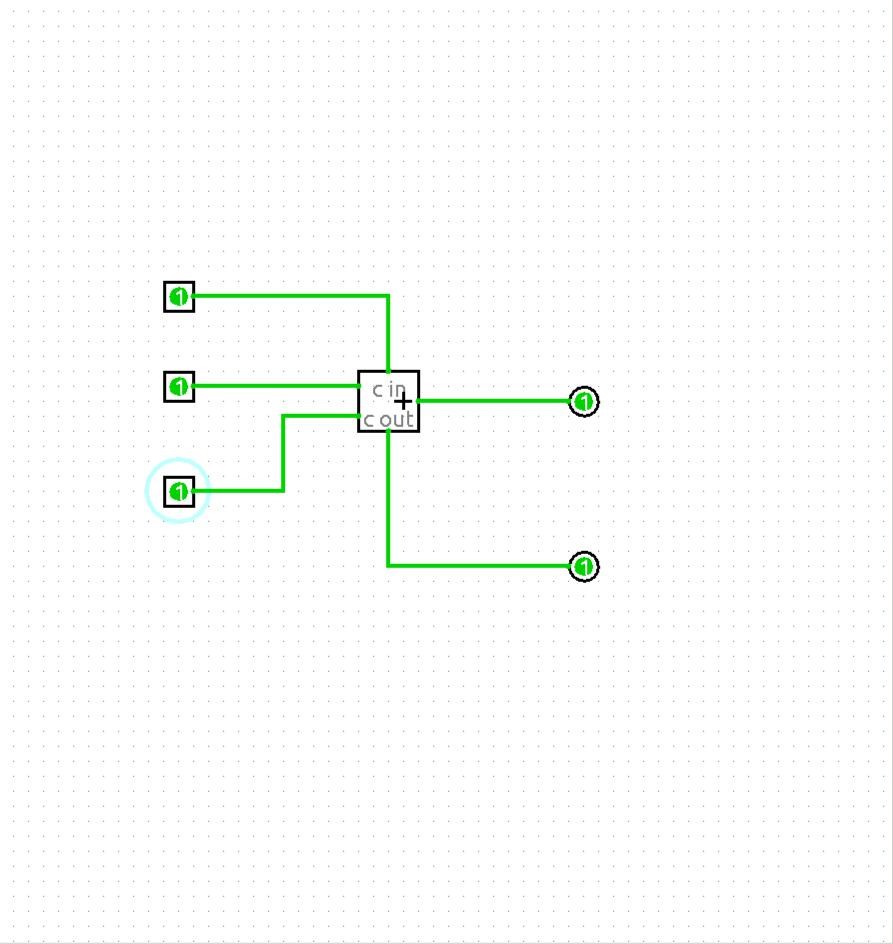
Theory:

Full Adder adds 3 inputs and produces 2 outputs.

The first 2 inputs are A and B third input is an input carry as C-IN.

The Output carry ia assigned as C-OUT and normal output is assigned as S which is SUM.

Output :



Performance date : 15.03.2022

# EXPERIMENT NO. 3

Aim : To study MUX and DeMUX

Tool used: Logisim

Theory :

Multiplexer is a data selector which takes several inputs and gives a single output.

Demultiplexer is a data distributor which takes a single input and gives several outputs.

MUX and DeMUX both are used in communication systems to carry out the process of data transmission.

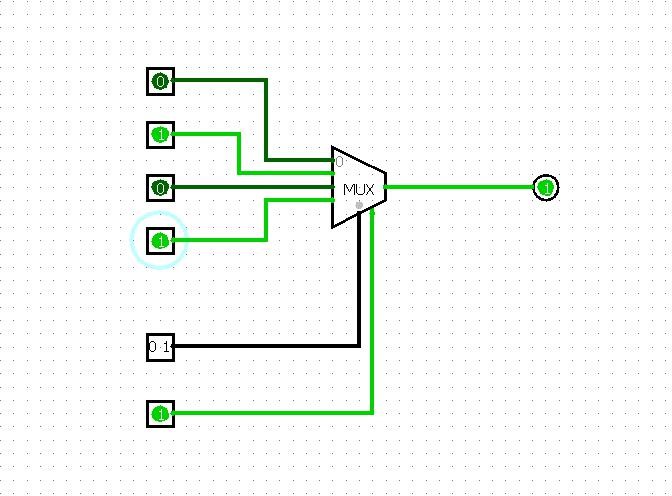
A multiplexer is a circuit used to select and route any one of the several input signals to a single output.

Demultiplexer receive the output data of Multiplexer(as a receiver) and convert back them to the original form then.

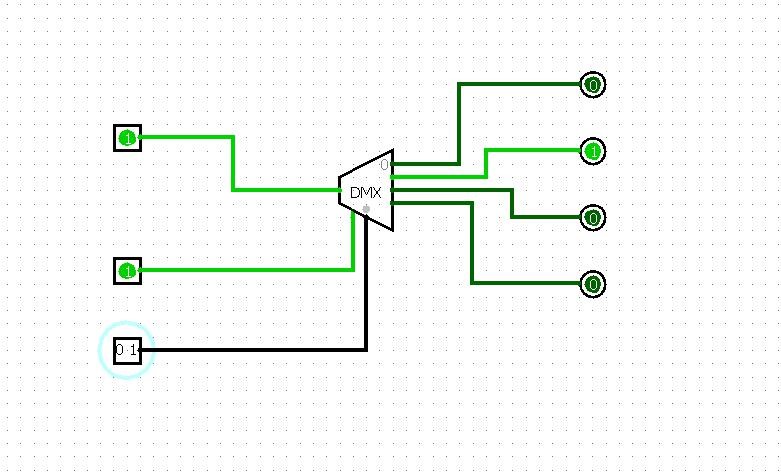
Demultiplexer and Multiplxer both are used in communication systems to carry multiple data signals.

Output:

MUX :



DeMUX



Performance date : 20.03.2022

# EXPERIMENT NO. 4

Aim: Study of PC mother board

Theory :

Northbridge is located on the northern or upper portion of the Motherboard. The main function of North bridge is to manage the communications between CPU and patrs of Motherboard.

Southbridge is located in the southern or lower portion of motherboard. The main function of South Bridge is to control IO functioning.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Components used in computer systems are :

Processor – Part of the computer that executes program instructions to process data, and handles main memory and i/o operations.

Main memory – Stores data and instructions while they are being processed

I/O Controllers – Accesses other components such as screen and secondary storage, which is used to permenently store data such as the operating system and user files.

Performance date : 04.02.2022

# EXPERIMENT NO. 5

Aim: Arithmetic operation using EMU8086

Program to perform :

addition of 2 16 bit numbers   
subtraction of 2 16bit numbers   
multiplication of 2 16 bit numbers

16/8 division

Theory :

Describe with examples

Any 5 data transfer instruction of 8086

MOV

Used to copy the byte or word from the provided source to the provided destination. Eg. mov ax, bx

POP

Used to get a word from the top of the stack to the provided location.

Eg. POP cx

PUSH

Used to put a word at the top of the stack.

Eg. POP cx

IN

Used to read a byte or word from the provided port to the accumulator.

Eg. IN AX, 1326H IN AL, DX

OUT

Used to send out a byte or word from the accumulator to the provided port.

Eg. OUT 1326H, AL

OUT DX, AX

Any 5 arithmetic instructions of 8086.

ADD :-

Adds the given byte to byte

Eg. mov ax, 2000h mov bx, 5000h add ax, bx

SUB :-

Subtracts the given byte from byte

Eg.mov ax, 54ABh mov bx, 2000h sub ax, bx

MUL :-

Multiply’s the unsigned word by byte

Eg.mov al, 02h mov bl, 05h mul bl

DIV :-

Divides the unsigned word by byte

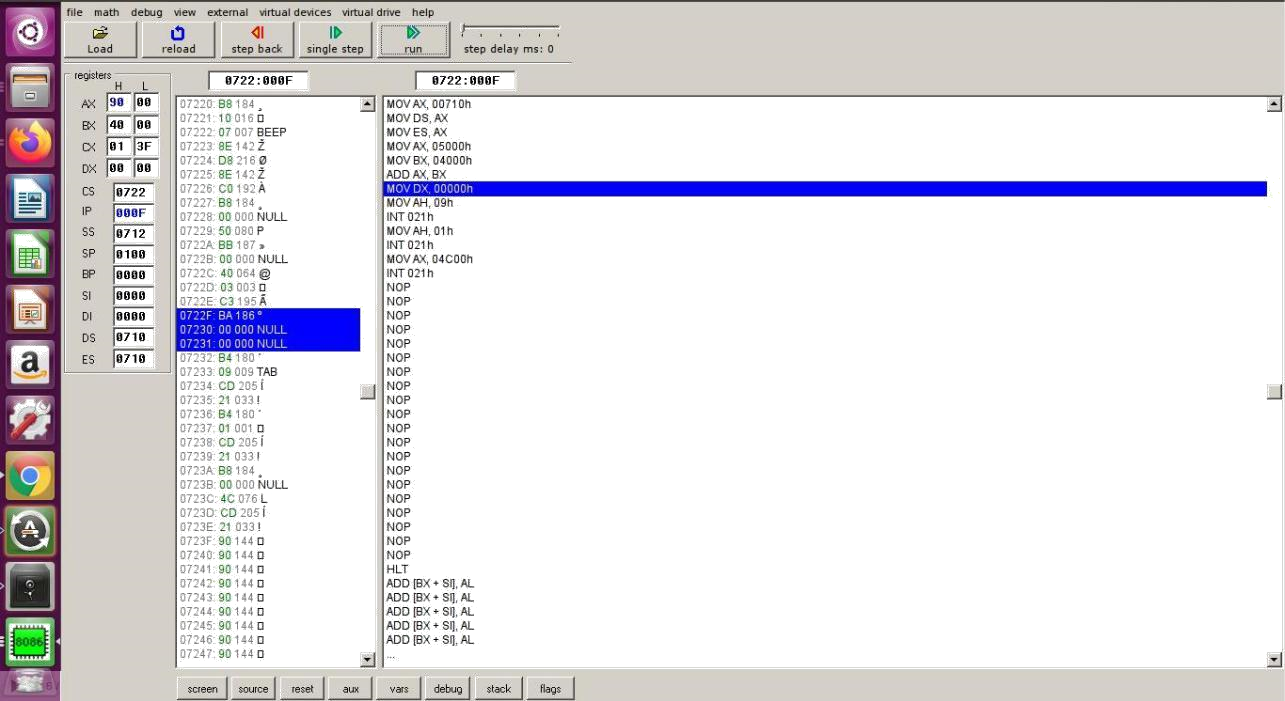
Eg.mov ax, 0100h mov bl, 05h

div bl

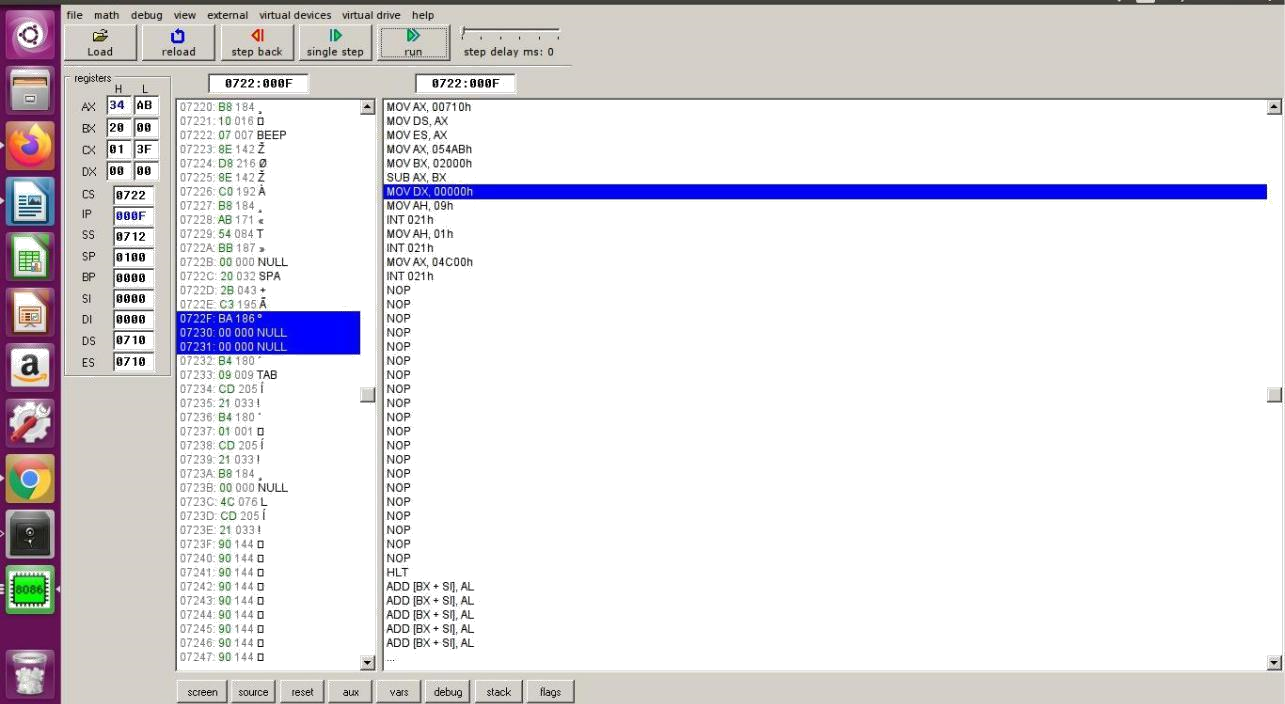
INC :- Increments the given byte by 1

Output

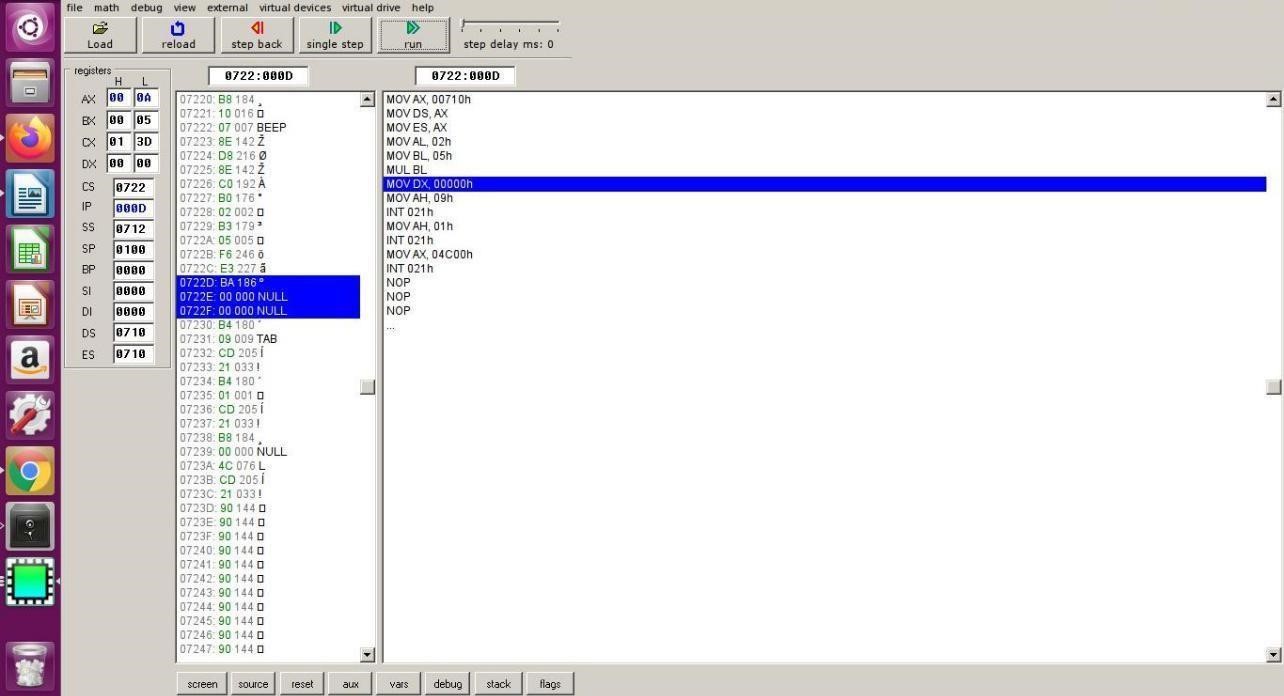
Add



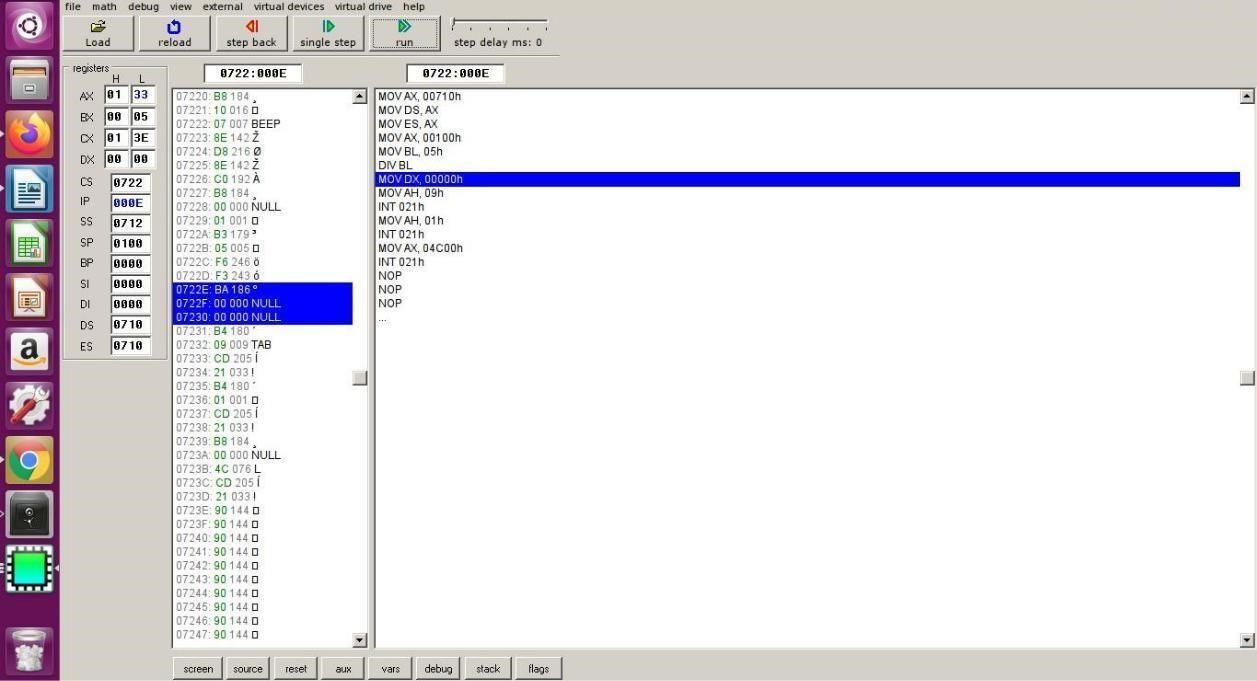
Subtraction



Multiplication



Division



Performance date : 29.03.2022

# EXPERIMENT NO. 6

Aim:- write a program to calculate the number of zeros and ones in an 8 bit number.

Theory

Describe with examples:-

Any 5 logic instructions of 8086

AND D, S D=DANDS AND AX, 0010

OR D, S D=DORS ORAX,BX

NOT D D = NOT of D NOT AL

XOR D, S D=DXORS XOR AL, BL

TEST D, S performs bit-wise AND operation and affects the flag registor TEST [0250], 06

Any 5 branch control instructions of 8086

JC Jump if Carry C=1

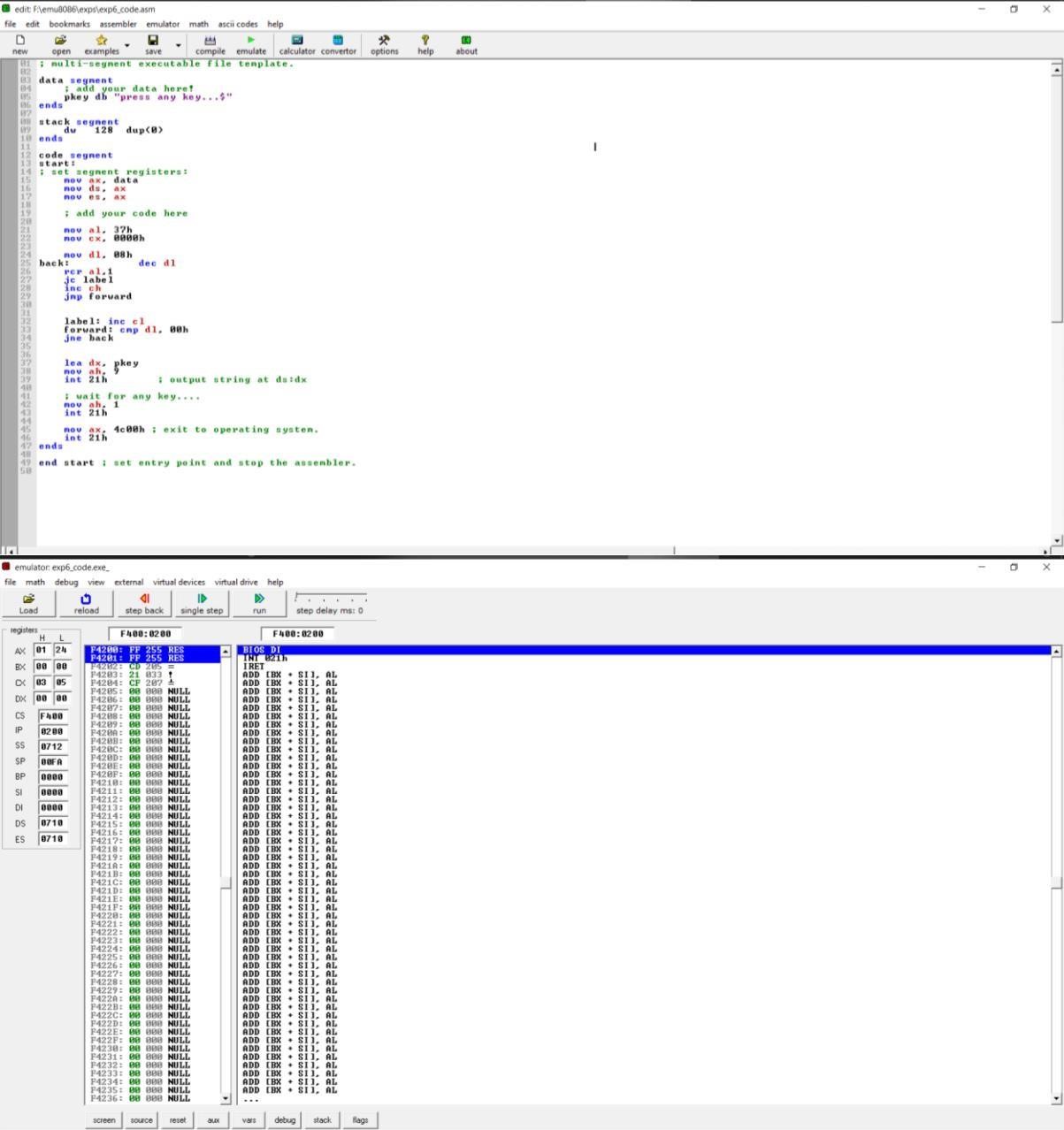
JNC Jump if Not Carry C=0

JCXZ Jump if the CX register=0 CX=0

JE/JZJump if Equal/Jump if Zero Z=1

JG/JNLE Jump if Greater/Jump if Not Less Than or Equal ((S xor O) or Z) = 0

Output:-



Performance date : 04.02.2022

# EXPERIMENT NO. 7

Aim:

Write a program to find the largest and smallest number out of 5 numbers

Theory

Describe with examples

Any 5 string instructions of 8086:-

REP instruction repeat the given instruction till CX != 0 REP MOVSB

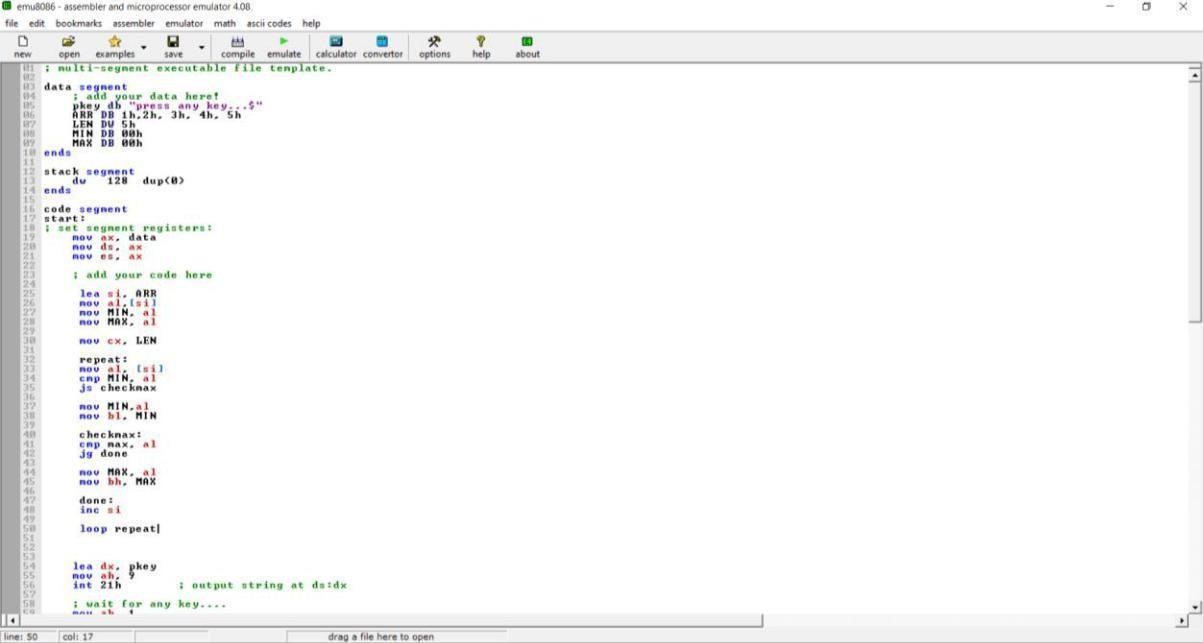
REPE instruction repeat the given instruction while CX = 0 REPE

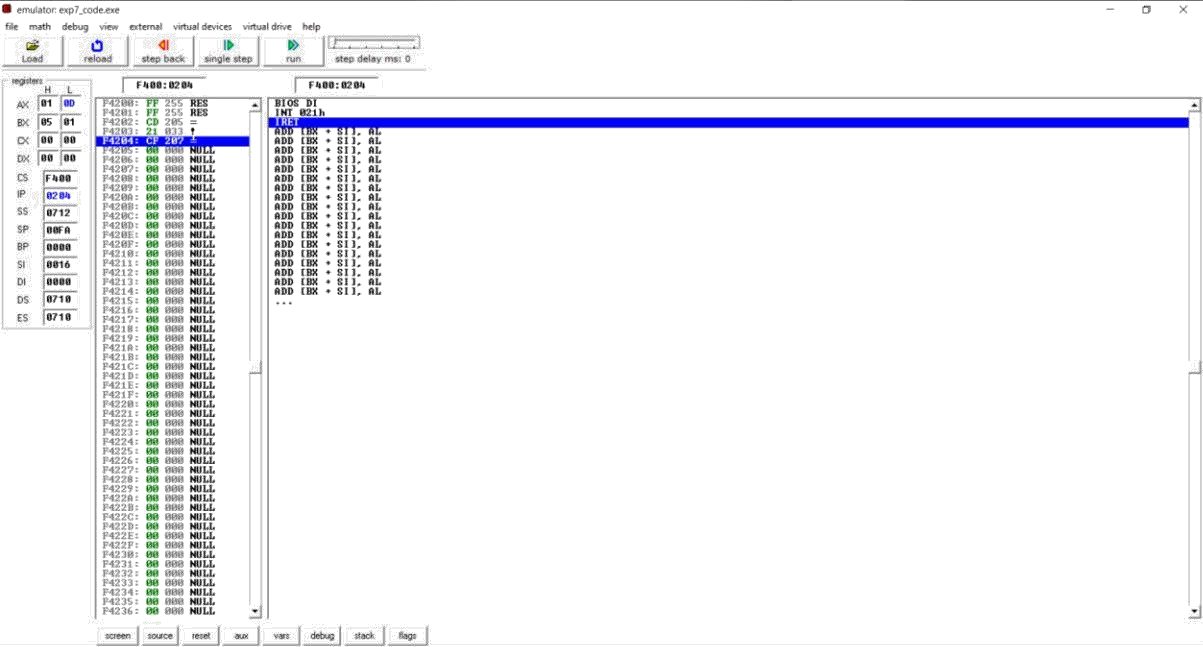
REPZ instruction repeat the given instruction while ZF = 1 REPZ

REPNE instruction repeat the given instruction while CX != 0 REPNE

REPNZ instruction repeat the given instruction while ZF = 0 REPNZ

Output





Performance date : 04.04.2022

EXPERIMENT NO. 8

**Aim:**

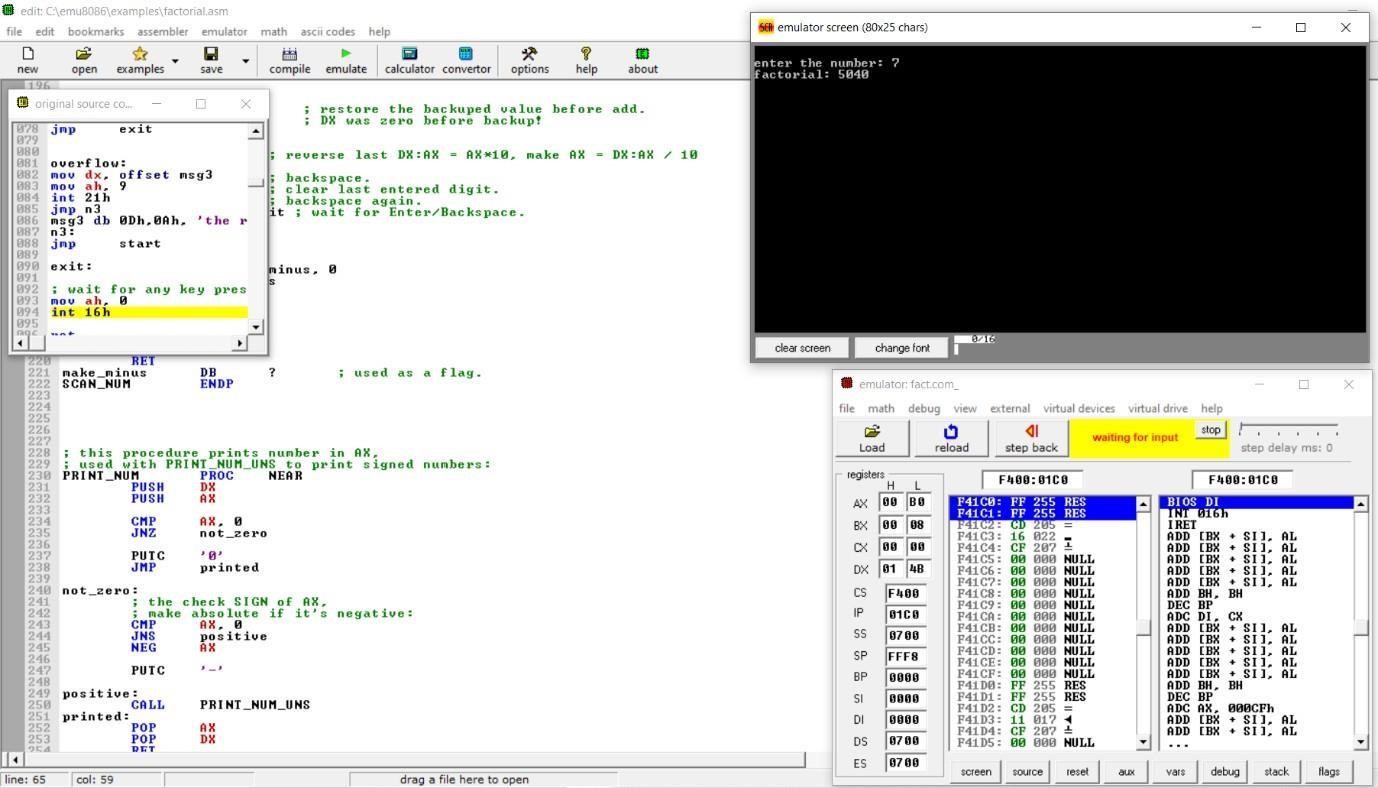
To calculate a factorial of a number.

**Theory:**

Distinguish between procedures and macros

|  |  |
| --- | --- |
| **Procedures** | **Macros** |
| Procedure contains a set of instructions which can be called repetitively which can perform a specific task. | Macro definition contains a set of instruction to support modular programming. |
| It is used for large set of instructions mostly more than ten instructions | It is used for small set of instructions mostly less than ten instructions |
| In case of procedure memory requirement is less. | In case of macro memory requirement is high |
| CALL and RET  instruction/statements are required in procedure | CALL and RET  instruction/statements are not required in macro |
| Assembler directive PROC is used to define procedure and assembler directive | Assembler directive MACRO is used to define macro and assembler directive |
| ENDP is used to indicate the body is over. | ENDM is used to indicate the body is over |
| Execution time of procedures is high as it executes slower than macro. | Execution time of macro is less than it executes faster than procedure. |
| Overhead time takes place during calling procedure and returning control to calling program | Overhead time does not take place as there is no calling and returning |
| Here machine code is created only once, it is generated only once when the procedure is defined. | Here machine code is created multiple times as each time machine code is generated when |
|  | macro is called. |

**Output:**



Performance date : 04.04.2022

# EXPERIMENT NO. 9

**Aim:**

To calculate a whether a given string is a palindrome or not

**Theory:**

Write a short note on parameter passing in procedures

**Answer :**

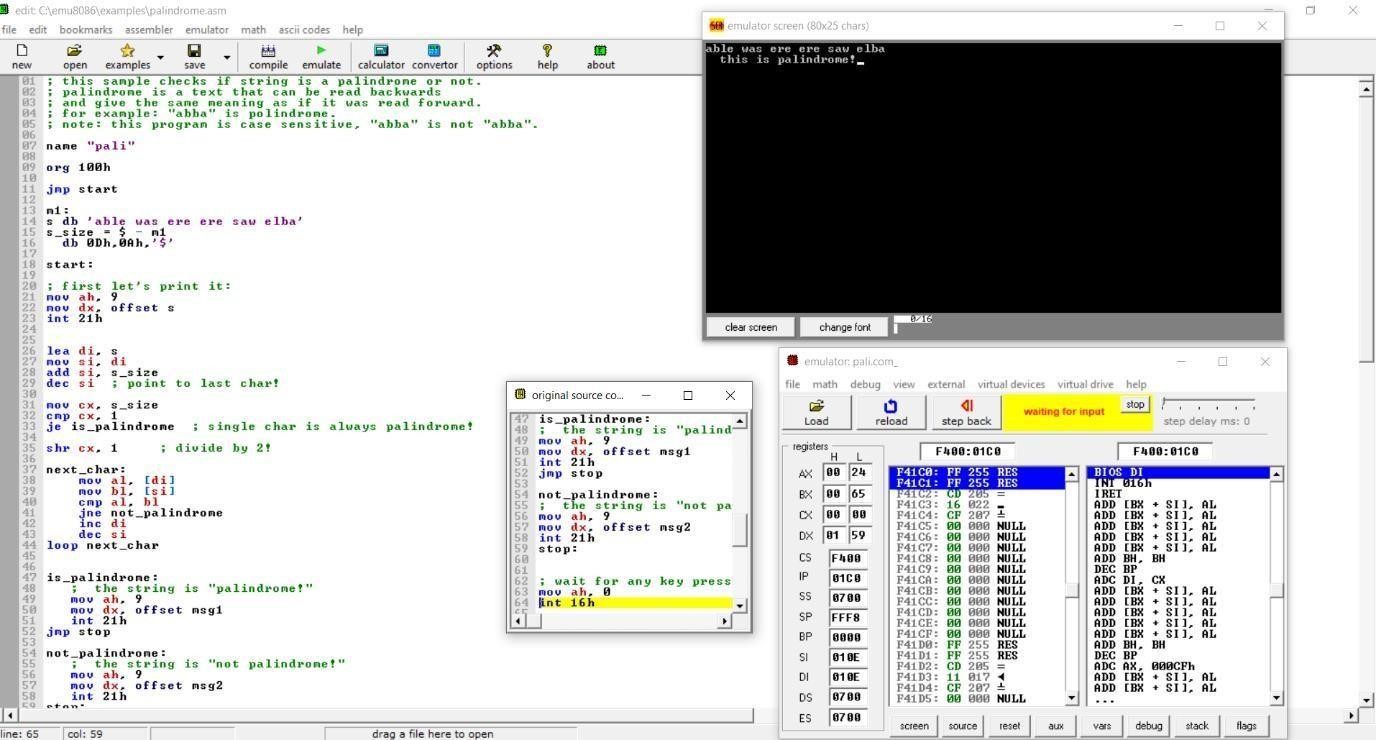
When writing procedures it is usually necessary to pass information to the procedure to customize its operation. In high-level languages this information is in the form of parameters that are included as part of the procedure call. In assembler, there are various techniques to pass information.

The simplest technique is to simply store the information in various registers.

IOASM uses this technique, where the information is stored in the AX register. The advantage of using this technique is that information in the registers can be immediately used, and the code is simpler and faster. The disadvantage is that the registers restrict the number of parameters and they become tied up once a value is stored in a particular register. Also, once a procedure is written to use a particular register, every calling routine must set the parameter in that same register, irrespective of whether it is in use or not.

The more general technique is to use the stack to store parameters. The advantage is that there are no restrictions to size or register usage. The disadvantage is that the code becomes a bit more complex.

**Output:**



END

\_