Name:	Rahul Yadav
Roll No:	76
Class/Sem:	SE/IV
Experiment No.:	2B
Title:	Program for calculating factorial using assembly language
Date of Performance:	24/01/2024
Date of Submission:	31/01/2024
Marks:	
Sign of Faculty:	

Aim: Program to calculate the Factorial of a number

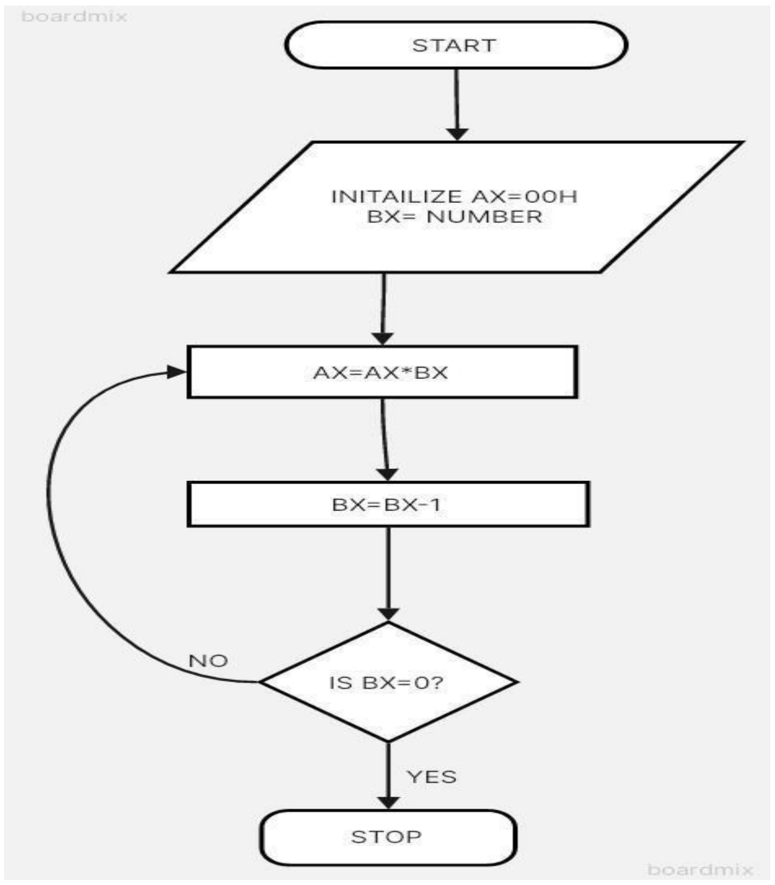
Theory:

To calculate the factorial of any number, we use MUL instruction. Here, initially, we initialize the first register by value 1. The second register is initialized by the value of the second register. After multiplication, decrement the value of the second register and repeat the multiplying step till the second register value becomes zero. The result is stored in the first register.

Algorithm:

1. Start.
2. Set AX=01H, and BX with the value whose factorial we want to find.
3. Multiply AX and BX.
4. Decrement BX=BX-1.
5. Repeat steps 3 and 4 till BX=0.
6. Stop.

Flowchart:

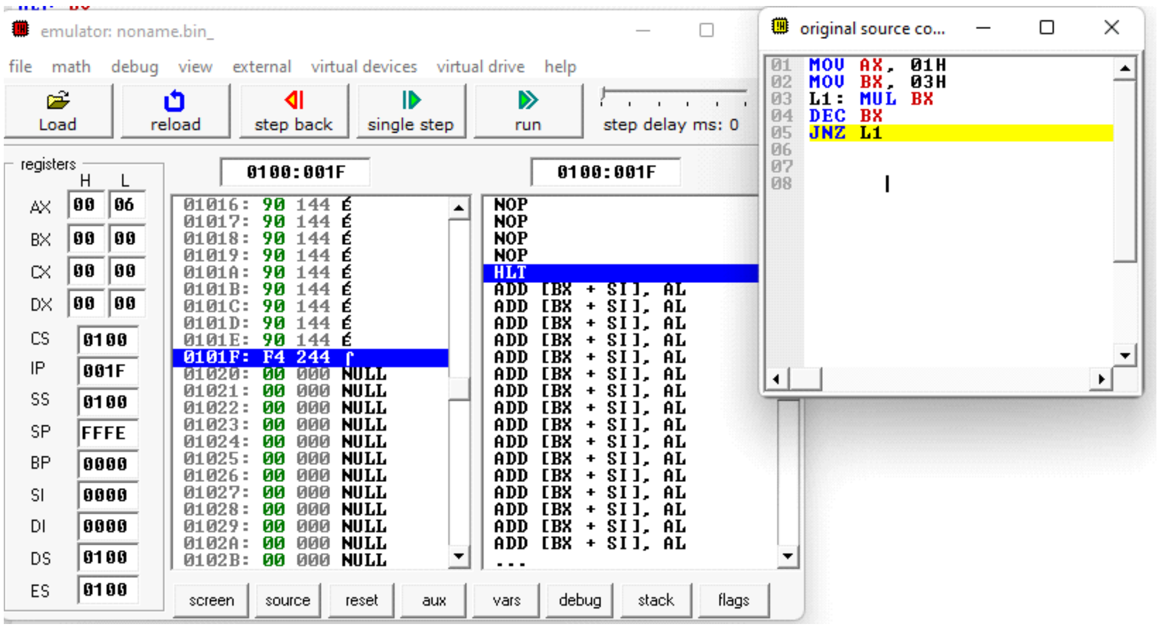


Code:

Using Factorial

```
MOV AX, 01H

MOV BX, 03H

L1: MUL BX

DEC BX

JNZ L1
```

Output:



emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
original source co...
screen source reset aux vars debug stack flags

```
registers
     H   L
AX  00  06
BX  00  00
CX  00  00
DX  00  00
CS  0100
IP  001F
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:001F                      0100:001F
01016: 90 144 É                NOP
01017: 90 144 É                NOP
01018: 90 144 É                NOP
01019: 90 144 É                NOP
0101A: 90 144 É                HLT
0101B: 90 144 É                ADD [BX + SI], AL
0101C: 90 144 É                ADD [BX + SI], AL
0101D: 90 144 É                ADD [BX + SI], AL
0101E: 90 144 É                ADD [BX + SI], AL
0101F: F4 244 ſ                ADD [BX + SI], AL
01020: 00 000 NULL             ADD [BX + SI], AL
01021: 00 000 NULL             ADD [BX + SI], AL
01022: 00 000 NULL             ADD [BX + SI], AL
01023: 00 000 NULL             ADD [BX + SI], AL
01024: 00 000 NULL             ADD [BX + SI], AL
01025: 00 000 NULL             ADD [BX + SI], AL
01026: 00 000 NULL             ADD [BX + SI], AL
01027: 00 000 NULL             ADD [BX + SI], AL
01028: 00 000 NULL             ADD [BX + SI], AL
01029: 00 000 NULL             ADD [BX + SI], AL
0102A: 00 000 NULL             ADD [BX + SI], AL
0102B: 00 000 NULL             ...
```

```
MOV AX, 01H
MOV BX, 03H
L1: MUL BX
DEC BX
JNZ L1
```

<u>Conclusion:</u>

Utilizing the MUL instruction, the program efficiently computes the factorial of a given number by iteratively multiplying the current result with decreasing values until reaching zero. This approach demonstrates the power of assembly language in implementing complex mathematical operations using simple instructions.

1.Explain shift instructions.

In the Intel 8086 assembly language, shift instructions are used to shift the bits of a binary number to the left or right. These instructions are handy for manipulating binary data and are frequently used in tasks such as multiplication, division, and bit manipulation.

Here are the commonly used shift instructions in 8086 assembly language:

SHL (Shift Left) / SAL (Arithmetic Shift Left):

SHL and SAL are synonymous in 8086 assembly.
Syntax: SHL/SAL destination, count
Description: This instruction shifts all bits in the destination operand (register or memory) to the left by the number of bits specified in the count operand. Zeros are shifted into the low-order bits, and the high-order bits are discarded. The carry flag is set to the value of the last bit shifted out.

Example: SHL AX, 1 shifts the contents of the AX register one bit to the left.
SHR (Shift Right) / SAR (Arithmetic Shift Right):

SHR performs a logical shift right, while SAR performs an arithmetic shift right.
Syntax: SHR/SAR destination, count
Description: This instruction shifts all bits in the destination operand to the right by the number of bits specified in the count operand. For SHR, zeros are shifted into the high-order bits, and the low-order bits are discarded. For SAR, the sign bit is shifted into the high-order bits, preserving the sign of the original value. The carry flag is set to the value of the last bit shifted out.
Example: SHR BX, 1 shifts the contents of the BX register one bit to the right.
These shift instructions are fundamental for various bitwise operations and are extensively used in

low-level programming, especially in systems programming and device driver development. They enable efficient manipulation of binary data at the bit level, providing a high degree of control over data representation and processing

2. Explain rotate instructions

In 8086 assembly language, rotate instructions are used to rotate the bits of a binary number to the left or right, similar to shift instructions. However, rotate instructions differ in that the bits that are shifted out on one end are rotated back in on the other end. This circular rotation of bits is useful in certain types of data manipulation and cryptography algorithms.

Here are the rotate instructions commonly used in 8086 assembly language:

RCL (Rotate through Carry Left):

Syntax: RCL destination, count
Description: This instruction rotates all bits in the destination operand (register or memory) to the left by the number of bit positions specified in the count operand. The carry flag is rotated into the least significant bit, and the most significant bit is rotated into the carry flag. This allows for circular rotation of the bits through the carry flag.
Example: RCL AX, 1 rotates the contents of the AX register one bit to the left through the carry flag.
RCR (Rotate through Carry Right):

Syntax: RCR destination, count
Description: This instruction rotates all bits in the destination operand to the right by the number of bit positions specified in the count operand. The carry flag is rotated into the most significant bit, and the least significant bit is rotated into the carry flag. Similar to RCL, this allows for circular rotation of the bits through the carry flag.
Example: RCR BX, 1 rotates the contents of the BX register one bit to the right through the carry flag.
ROL (Rotate Left):

Syntax: ROL destination, count
Description: This instruction rotates all bits in the destination operand to the left by the number of bit positions specified in the count operand. The most significant bit is rotated into the least significant bit, and the carry flag is set to the value of the bit shifted out of the most significant position.
Example: ROL CX, 1 rotates the contents of the CX register one bit to the left.
ROR (Rotate Right):