Name:	Yash Ravindra Kerkar
Roll No:	67
Class/Sem:	SE/IV
Experiment No.:	2B
Title:	Program for calculating factorial using assembly language
Date of Performance:	24/01/24
Date of Submission:	31/01/24
Marks:	
Sign of Faculty:	

Aim: Program to calculate the Factorial of a number.

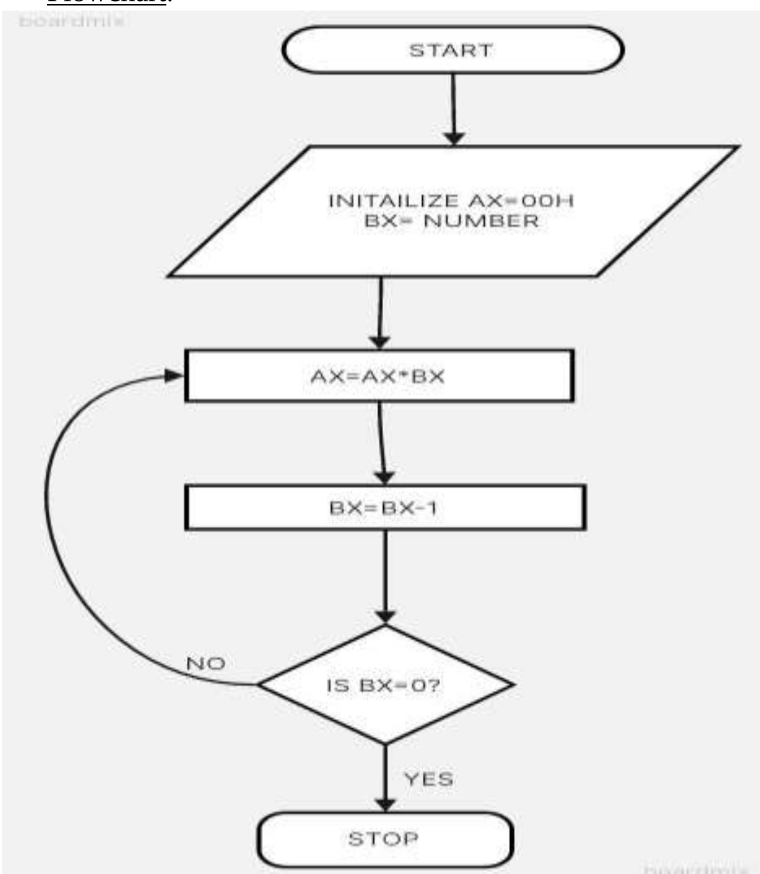
Theory:

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:



Program:

MOV AX, 0001H

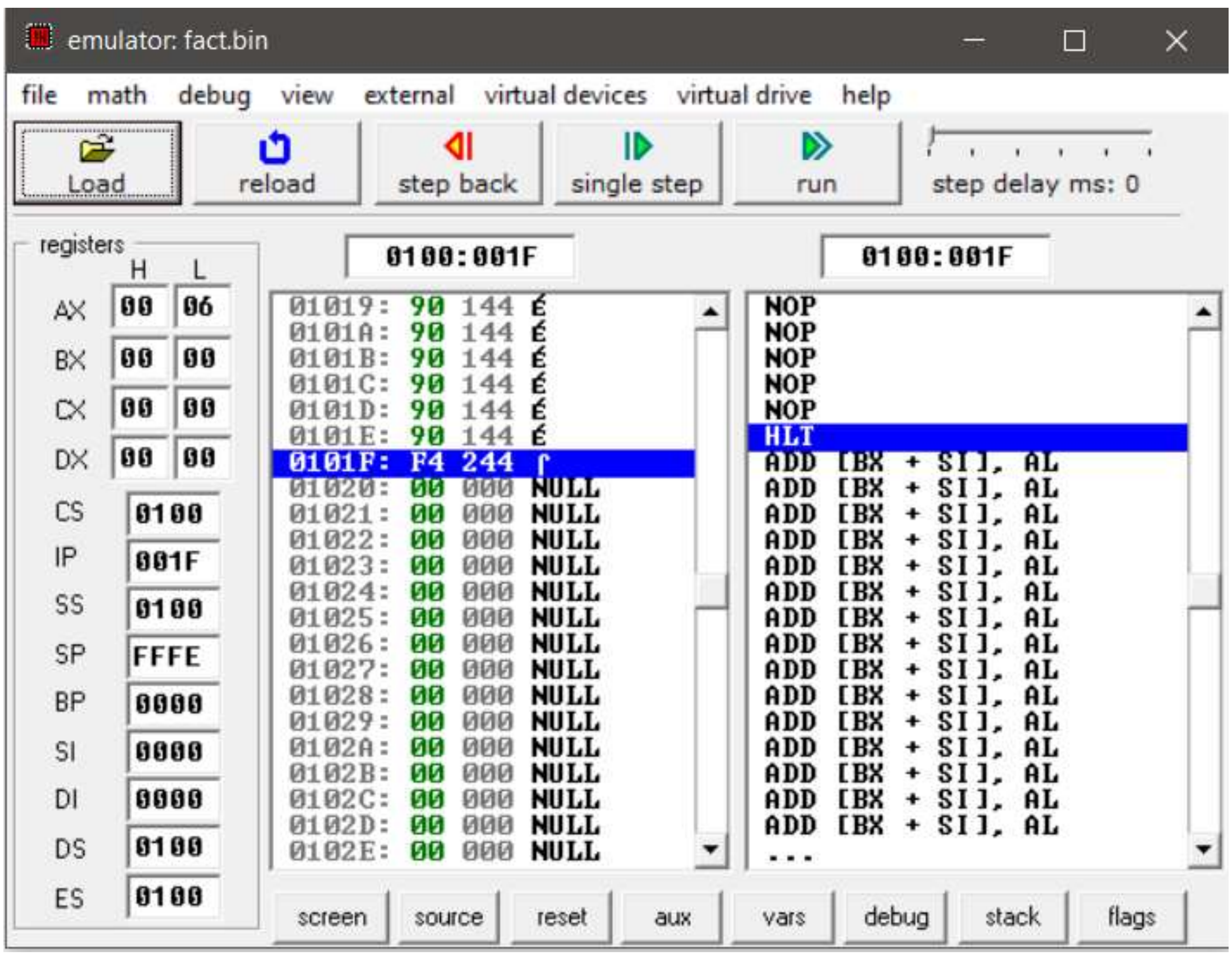
MOV BX, 0003H

L1: MUL BX

DEC BX

JNZ L1

Output:



Conclusion: We successfully completed the assembly language program designed to calculate the factorial of a number. By implementing a combination of MOV, MUL, DEC, and JNZ instructions, we effectively computed the factorial of a given value. This hands-on experience deepened our understanding of arithmetic operations and control flow mechanisms in assembly language programming.

1. **Explain shift instructions.**

Ans. **Arithmetic Shift Left (SHL/ SAL):** Syntax: `SHL destination, count` or `SAL destination, count`. Function: This instruction shifts the bits of the destination operand (register or memory location) to the left by the number of bit positions specified by the count operand. The empty bit positions on the right side are filled with zeros. This operation effectively multiplies the value by 2 for each shift left by one bit.

Arithmetic Shift Right (SHR): Syntax: `SHR destination, count`. Function: This instruction shifts the bits of the destination operand to the right by the number of bit positions specified by the count operand. The empty bit positions on the left side are filled with zeros. In signed numbers, the sign bit is retained, which means the leftmost bit (sign bit) is replicated to preserve the sign.
Logical Shift Right (SHR).

Logical Shift Right (SHR): Syntax: `SHR destination, count`. Function: Similar to arithmetic shift right, but in this case, the leftmost bit is always filled with zero, regardless of the sign bit of the original value. It's commonly used for unsigned integers.

Rotate Instructions (ROL, ROR, RCL, RCR): Rotate instructions are variations of shift instructions that shift the bits of the operand left or right, but the shifted out bits are circulated back to the opposite end. These instructions are useful for implementing circular buffers, encryption algorithms, and other tasks that require rotating bit patterns.

2. **Explain rotate instructions.**

Ans. Rotate Left (ROL): Syntax: `ROL destination, count`. Function: This instruction circularly shifts the bits of the destination operand (register or memory location) to the left by the number of bit positions specified by the count operand. The bits shifted out from the left side are rotated back and inserted into the right side. This operation effectively rotates the bits to the left.

Rotate Right (ROR): Syntax: `ROR destination, count`. Function: Similar to ROL, but rotates the bits to the right. The bits shifted out from the right side are rotated back and inserted into the left side.

Rotate Through Carry Left (RCL): Syntax: `RCL destination, count`. Function: This instruction performs a left rotation similar to ROL but includes the value of the Carry Flag (CF) in the rotation. The Carry Flag is shifted into the least significant bit (LSB) during the rotation, and the original LSB is shifted into the Carry Flag.

Rotate Through Carry Right (RCR): Syntax: `RCR destination, count`. Function: Similar to RCL, but rotates the bits to the right. The Carry Flag is shifted into the most significant bit.