Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL404**	**Course Name:**	**Microprocessor Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**2**
Title of the Experiment:	**A. Program to perform multiplication without using MUL** **B. Program for calculating factorial using assembly language**
Date of Performance:	**23/01/2025**
Date of Submission:	**30/01/2025**

Evaluation

Performance Indicator	**Max. Marks**	**Marks Obtained**
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	**Exceed Expectations (EE)**	**Meet Expectations (ME)**	**Below Expectations (BE)**
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Ms. Sweety Patil

Signature :

Date:

Aim: Program for multiplication without using the multiplication instruction.

Theory:

In the multiplication program, we multiply the two numbers without using the direct instructions MUL. Here we can successive addition methods to get the product of two numbers. For that, in one register we will take multiplicand so that we can add multiplicand itself till the multiplier stored in another register becomes zero.

ORG 100H:

It is a compiler directive. It tells the compiler how to handle source code. It tells the compiler that the executable file will be loaded at the offset of 100H (256 bytes.)

INT 21H:

The instruction INT 21H transfers control to the operating system, to a subprogram that handles I/O operations.

MUL: MUL Source.

This instruction multiplies an unsigned byte from some source times an unsigned byte in the AL register or an unsigned word from some source times an unsigned word in the AX register.

Source: Register, Memory Location.

When a byte is multiplied by the content of AL, the result (product) is put in AX. A 16-bit destination is required because the result of multiplying an 8-bit number by an 8-bit number can be as large as 16-bits. The MSB of the result is put in AH and the LSB of the result is put in AL.

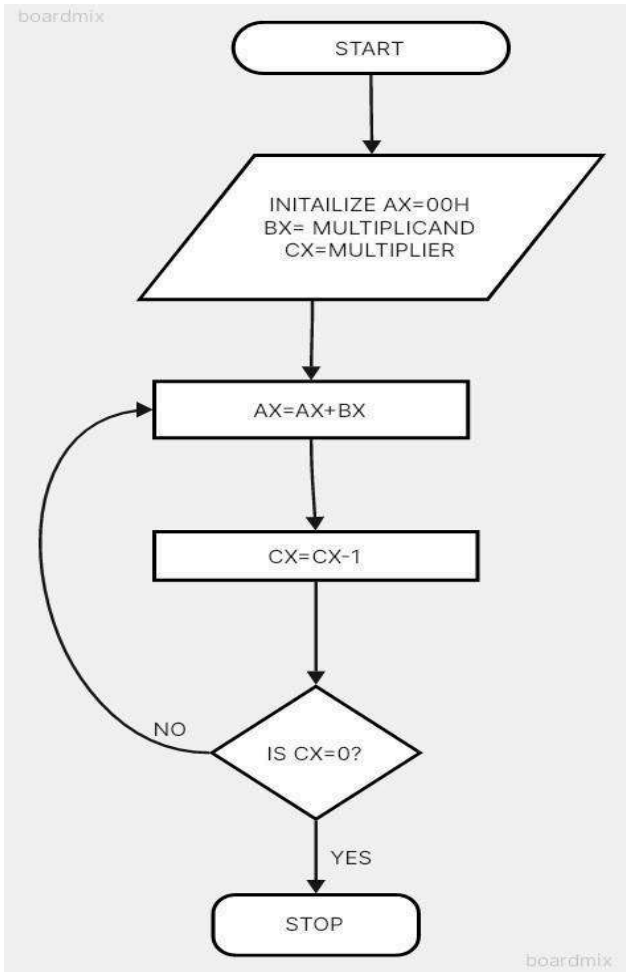
When a word is multiplied by the contents of AX, the product can be as large as 32 bits. The MSB of the result is put in the DX register and the LSB of the result is put in the AX register.

MUL BH; multiply AL with BH; result in AX.

Algorithm:

1. Start.
2. Set AX=00H, BX= Multiplicand, CX=Multiplier 3 Add the content of AX and BX.
4. Decrement content of CX.
5. Repeat steps 3 and 4 till CX=0.
6. Stop.

Flowchart:



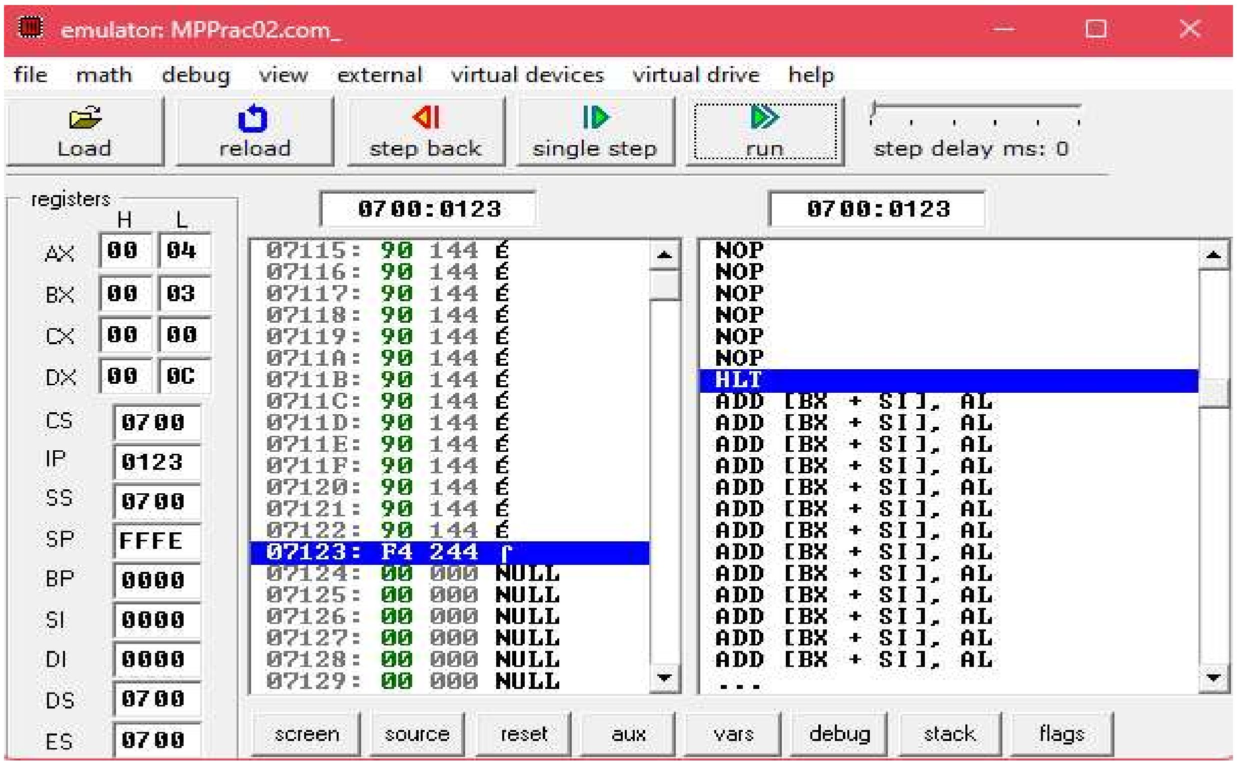
Code:

A)MULTIPLICATION WITHOUT USING MUL

```
MOV AX,0004H
MOV BX,0003H
MOV CX,BX
MOV DX,0

LOOPSTART:
ADD DX, AX
LOOP LOOPSTART
```

Output:


emulator: MPPrac02.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
registers
     H   L
AX   00  04
BX   00  03
CX   00  00
DX   00  0C
CS   0700
IP   0123
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0700:0123
07115: 90 144 É
07116: 90 144 É
07117: 90 144 É
07118: 90 144 É
07119: 90 144 É
0711A: 90 144 É
0711B: 90 144 É
0711C: 90 144 É
0711D: 90 144 É
0711E: 90 144 É
0711F: 90 144 É
07120: 90 144 É
07121: 90 144 É
07122: 90 144 É
07123: F4 244 ſ
07124: 00 000 NULL
07125: 00 000 NULL
07126: 00 000 NULL
07127: 00 000 NULL
07128: 00 000 NULL
07129: 00 000 NULL
```

```
0700:0123
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

Conclusion:

1. Explain data transfer instructions.
   Data transfer instructions in computer architecture are commands used to move data between registers, memory, or input/output devices. These instructions enable the copying, storing, or loading of data, allowing the processor to access and manipulate information efficiently during program execution. Examples include `MOV` or `LOAD`.

2. Explain Arithmetic instructions.

   Arithmetic instructions perform mathematical operations on data, such as addition, subtraction, multiplication, and division. These instructions are used to modify the contents of registers or memory locations. Examples include `ADD`, `SUB`, `MUL`, and `DIV`, which help in calculating values during program execution.

Aim: Program to calculate the Factorial of a number.

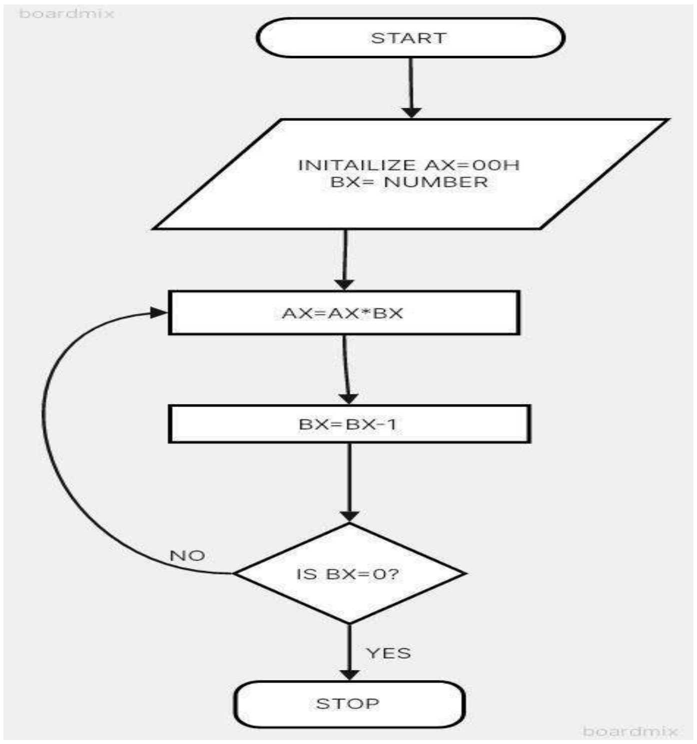
Theory:

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:



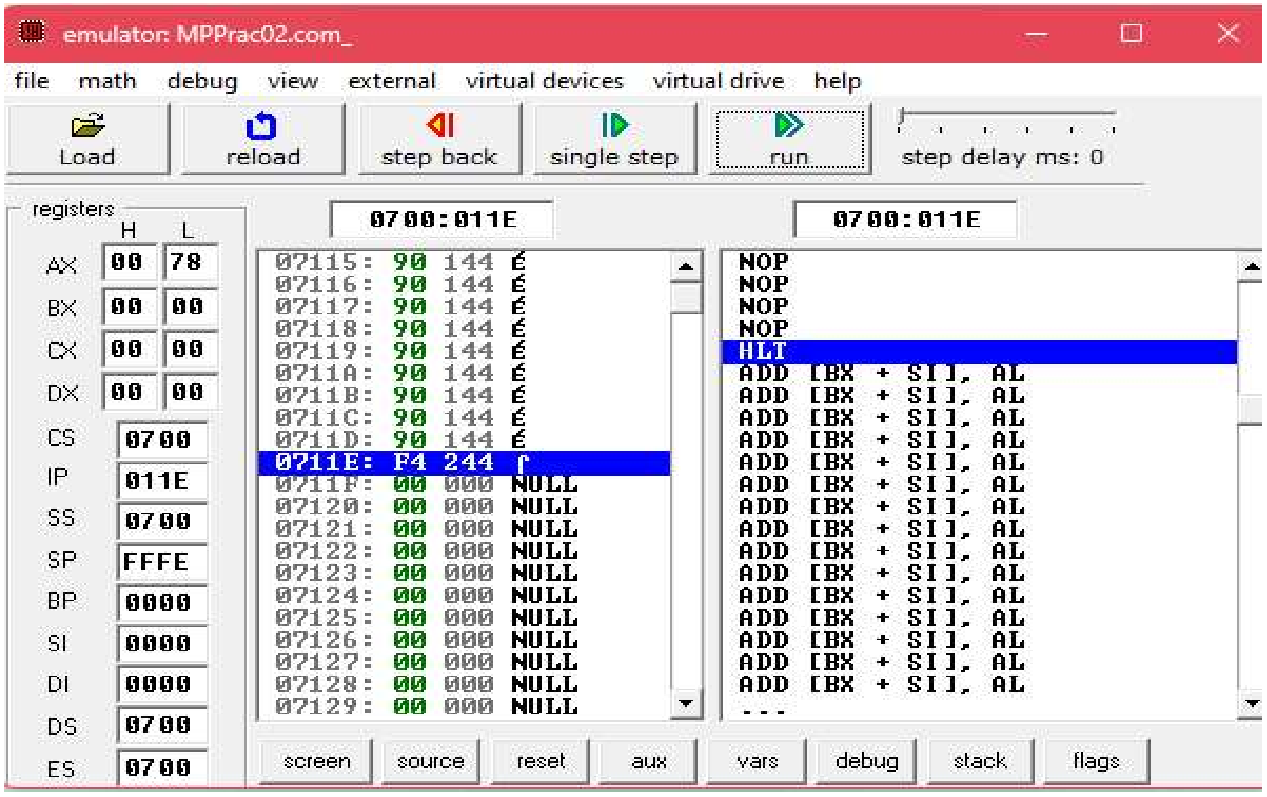
Code:

B)CALCULATING FACTORIAL USING ASSEMBLY LANGUAGE

```
MOV AX,0005H
MOV BX,AX
DEC BX

FACT:
MUL BX
DEC BX
JNZ FACT
```

Output:



```
registers
     H    L
AX   00   78
BX   00   00
CX   00   00
DX   00   00
CS   0700
IP   011E
SS   0700
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0700:011E                    0700:011E

07115:  90 144 É             NOP
07116:  90 144 É             NOP
07117:  90 144 É             NOP
07118:  90 144 É             NOP
07119:  90 144 É             HLT
0711A:  90 144 É             ADD [BX + SI], AL
0711B:  90 144 É             ADD [BX + SI], AL
0711C:  90 144 É             ADD [BX + SI], AL
0711D:  90 144 É             ADD [BX + SI], AL
0711E:  F4 244 ⌠             ADD [BX + SI], AL
0711F:  00 000 NULL          ADD [BX + SI], AL
07120:  00 000 NULL          ADD [BX + SI], AL
07121:  00 000 NULL          ADD [BX + SI], AL
07122:  00 000 NULL          ADD [BX + SI], AL
07123:  00 000 NULL          ADD [BX + SI], AL
07124:  00 000 NULL          ADD [BX + SI], AL
07125:  00 000 NULL          ADD [BX + SI], AL
07126:  00 000 NULL          ADD [BX + SI], AL
07127:  00 000 NULL          ADD [BX + SI], AL
07128:  00 000 NULL          ADD [BX + SI], AL
07129:  00 000 NULL          ...
```

Conclusion:

In conclusion, calculating the factorial using assembly language involves utilizing basic operations like multiplication and loops. The process involves loading the number into a register, iterating through a loop, and multiplying the result by the current counter value until the desired factorial is achieved. Although more complex than higher-level languages, assembly language provides a deeper understanding of low-level operations and memory manipulation.

1. Explain shift instructions.
   Shift instructions manipulate the bits of a register or memory operand by shifting them to the left or right. These operations can be logical (filling with zeros) or arithmetic (preserving the sign bit). Common shift instructions include `SHL` (shift left) and `SHR` (shift right), which are used for tasks like multiplication or division by powers of 2.

2. Explain rotate instructions

   Rotate instructions shift bits in a register or memory operand but differ from shift instructions by rotating the bits around the ends. When the bits are shifted out, they are reintroduced at the opposite end. Common rotate instructions include `ROL` (rotate left) and `ROR` (rotate right), which are used for bit manipulation tasks, like encryption or circular buffer management.