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.

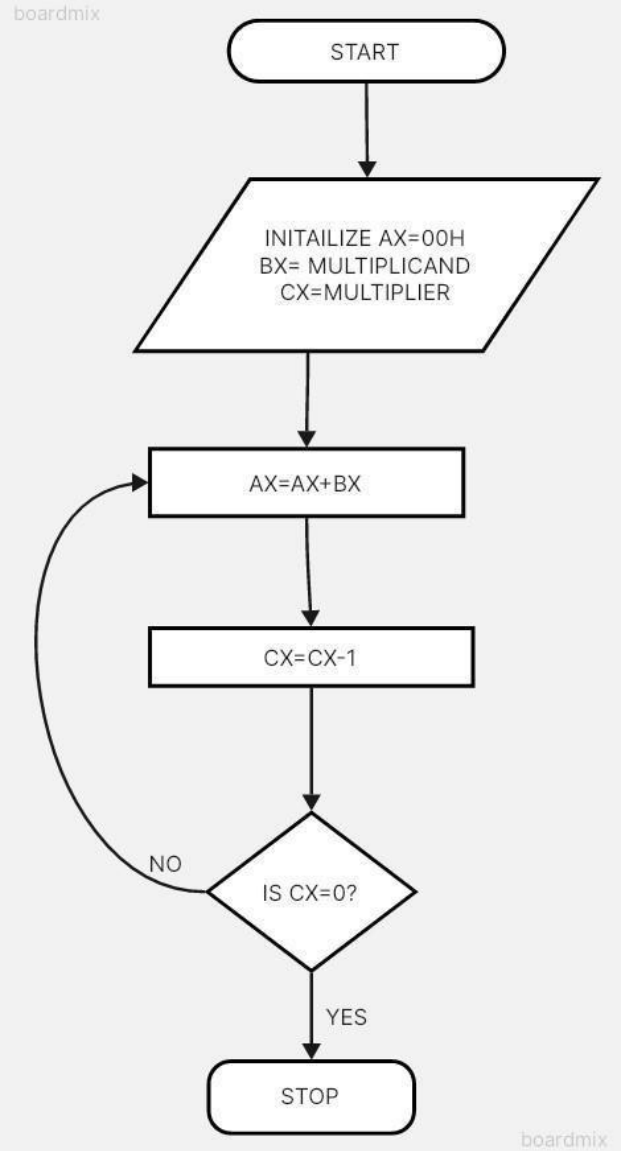
When a word is multiplied by the contents of AX, the product can be as large as 32 bits. The MSB of the result is put in the DX register and the LSB of the result is put in the AX register.

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.

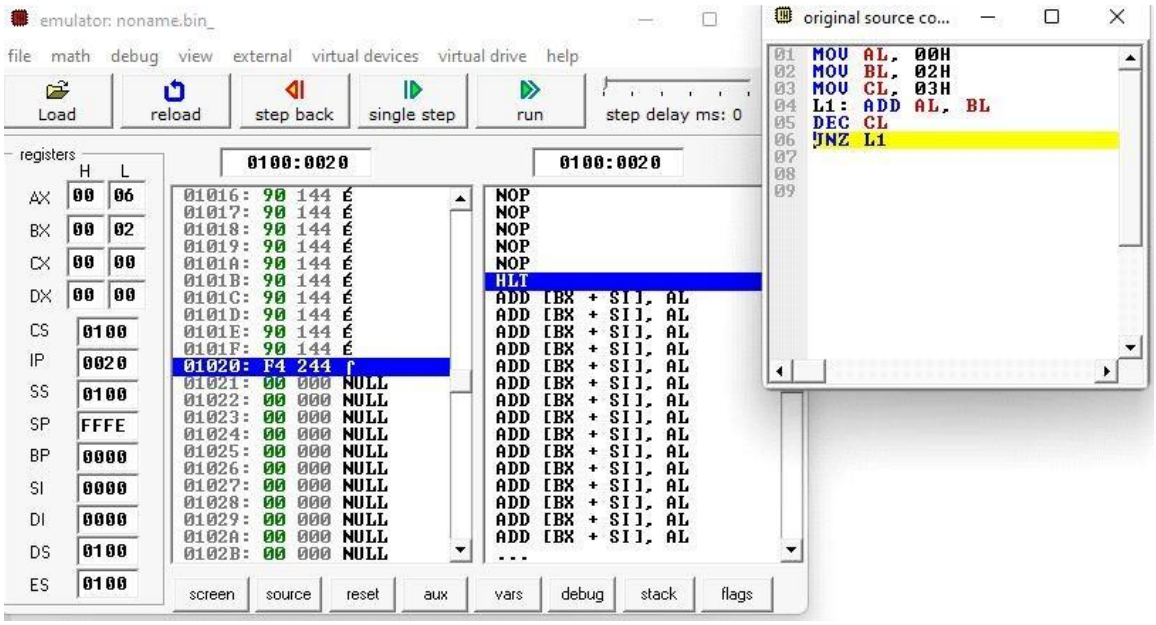
Flowchart:



Code:

```
MOV AL, 00H

MOV BL, 02H

MOV CL, 03H

L1: ADD AL, BL

DEC CL

JNZ L1
```

Output:



Conclusion:

By employing successive addition instead of the direct multiplication instruction, the program effectively computes the product of two numbers. This approach utilizes basic arithmetic operations to achieve multiplication, showcasing the versatility of assembly language in performing complex tasks through simple instructions and algorithms.

1.Explain data transfer instructions.

Data transfer instructions in the 8086 microprocessor facilitate the movement of data between registers, memory, and I/O devices. Key instructions include:

MOV: Copies data from a source operand to a destination operand.
XCHG: Swaps the contents of two operands.
PUSH: Stores data onto the stack.
POP: Retrieves data from the stack.
LEA: Loads the effective address of a memory operand into a register.
LDS and LES: Load pointer instructions used for loading segment and offset values from memory into segment and pointer registers.
LAHF and SAHF: Load and store flags instructions used for manipulating the flags register.
These instructions are essential for manipulating data within the processor and interfacing with external memory and devices in assembly language programming.

2.Explain Arithmetic instructions.

Arithmetic instructions in the 8086 microprocessor perform mathematical operations on data stored in registers and memory. Key arithmetic instructions include:

ADD: Adds the source operand to the destination operand, storing the result in the destination.
SUB: Subtracts the source operand from the destination operand, storing the result in the destination.
INC: Increments the value of the destination operand by one.
DEC: Decrements the value of the destination operand by one.
MUL: Multiplies the source operand by the accumulator (AL or AX), storing the result in AX or DX:AX.
IMUL: Signed multiplication operation, similar to MUL but for signed numbers.
DIV: Divides the contents of DX:AX by the source operand, storing the quotient in AX and the remainder in DX.
IDIV: Signed division operation, similar to DIV but for signed numbers.
These instructions enable the 8086 processor to perform various arithmetic operations efficiently, facilitating mathematical computations in assembly language programming.