Name:	Yash Ravindra Kerkar
Roll No:	67
Class/Sem:	SE/IV
Experiment No.:	2A
Title:	Program to perform multiplication without using MUL instruction
Date of Performance:	24/01/24
Date of Submission:	31/01/24
Marks:	
Sign of Faculty:	

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.

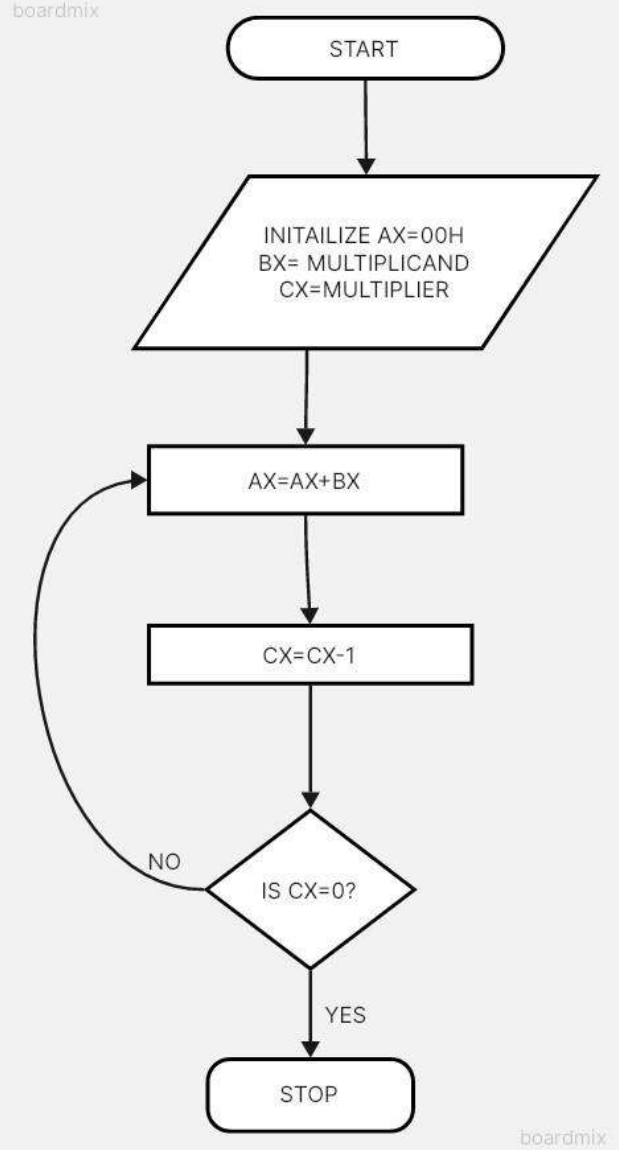
When a word is multiplied by the contents of AX, the product can be as large as 32 bits. The MSB of the result is put in the DX register and the LSB of the result is put in the AX register.

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.

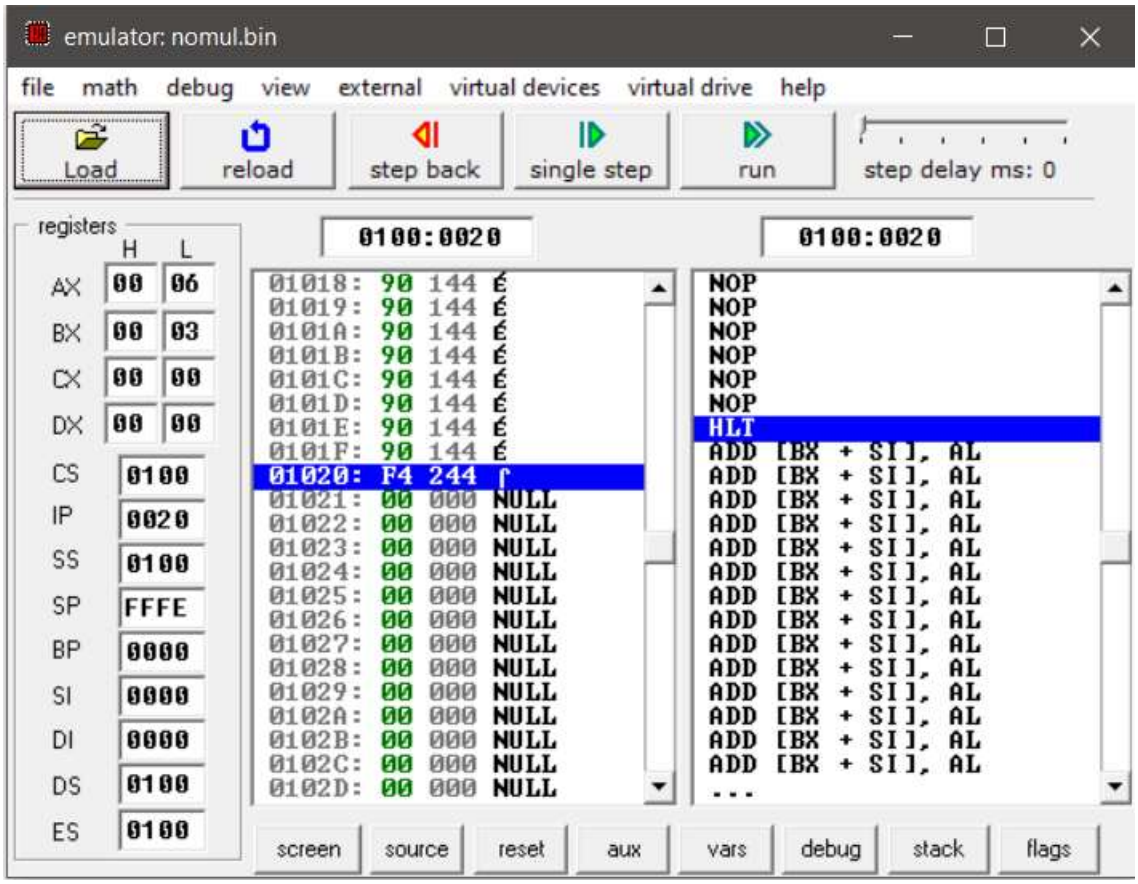
Flowchart:


boardmix
START
INITAILIZE AX=00H
BX= MULTIPLICAND
CX=MULTIPLIER
AX=AX+BX
CX=CX-1
NO
IS CX=0?
YES
STOP
boardmix

Program:

```
MOV AL, 00H
MOV BL, 03H
MOV CL, 02H
L1: ADD AL, BL
    DEC CL
    JNZ L1
```

Output:



Conclusion: We successfully implemented a program to perform multiplication without using the MUL instruction in x86 assembly language. The program showcased an alternative approach to multiplication using repeated addition, leveraging fundamental arithmetic principles and control flow constructs.

1) Explain data transfer instructions.

Ans. MOV Instruction: MOV instruction is used to copy data from one location to another without altering the original data. Examples include the `MOV` instruction, which is widely used to transfer data between registers, memory locations, and immediate values. Syntax: `MOV destination, source.` Example: `MOV AX, BX` copies the contents of the BX register into the AX register.

Load and Store Instructions: Load instructions are used to transfer data from memory to a register, while store instructions move data from a register to memory. Examples include the `MOV` instruction when used to transfer data between registers and memory locations. Syntax: `MOV destination, source` Example: `MOV AX, [MemoryLocation]` loads the data stored at MemoryLocation into the AX register.

Push and Pop Instructions: Push instructions are used to store data onto the stack, while pop instructions retrieve data from the stack. Examples include the `PUSH` and `POP` instructions in x86 assembly language. Syntax: `PUSH source` and `POP destination` Example: `PUSH AX` pushes the contents of the AX register onto the stack, while `POP BX` pops the top value from the stack and stores it in the BX register.

Exchange Instructions: Exchange instructions swap the contents of two operands without using an intermediate location. Examples include the `XCHG` instruction in x86 assembly language. Syntax: `XCHG operand1, operand2` Example: `XCHG AX, BX` swaps the contents of the AX and BX registers.

String Instructions: String instructions are used to transfer blocks of data within memory, often used for operations on strings. Examples include `MOVSB`, `MOVSW`, `MOVSX`, and `MOVSD` instructions. Syntax: Varies depending on the specific instction and the direction of transfer (e.g., forward or backward). Example: `MOVSB` moves a byte from the source address to the destination address and adjusts the pointers accordingly.

2) Explain Arithmetic instructions.

Ans. Addition (ADD): The `ADD` instruction adds the value of the source operand to the destination operand and stores the result in the destination operand. Syntax: `ADD destination, source.` Example: `ADD AX, BX` adds the value of the BX register to the AX register and stores the result in AX.

Subtraction (SUB): The `SUB` instruction subtracts the value of the source operand from the destination operand and stores the result in the destination operand. Syntax: `SUB destination, source.` Example: `SUB AX, BX` subtracts the value of the BX register from the AX register and stores the result in AX.

Multiplication (MUL): The `MUL` instruction multiplies the value of the source operand by the value in the AX register and stores the result in AX (for 8-bit source operand) or DX:AX (for 16-bit source operand). Syntax: `MUL source`. Example: `MUL BX` multiplies the value in the BX register by the value in AX and stores the result in DX:AX.

Division (DIV): The `DIV` instruction divides the value in the DX:AX register pair (for 16-bit operands) by the value of the source operand and stores the quotient in AX and the remainder in DX. Syntax: `DIV source.` Example: `DIV BX` divides the value in DX:AX by the value in the BX register and stores the quotient in AX and the remainder in DX.

Increment (INC): The `INC` instruction increments the value of the specified operand by Syntax: `INC operand.` Example: `INC CX` increments the value in the CX register by 1.

Decrement (DEC): The `DEC` instruction decrements the value of the specified operand by 1. Syntax: `DEC operand.` Example: `DEC DX` decrements the value in the DX register by 1.