Name:	Viraj Oza
Roll No:	69
Class/Sem:	SE/IV
Experiment No.:	1
Title:	To perform basic arithmetic operations on 16-bit data.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Aim: Assembly Language Program to perform basic arithmetic operations (addition, subtraction, multiplication, and division) on 16-bit data.

Theory:

MOV: MOV Destination, Source.
The MOV instruction copies data from a specified destination. word or byte of data from a specified destination.
Source: Register, Memory Location, Immediate Number
Destination: Register, Memory Location

MOV CX, 037AH; Put immediate number 037AH to CX.
ADD: ADD Destination, Source.

These instructions add a number source to a number from some destination and put the result in the specified destination.

Source: Register, Memory Location, Immediate Number

Destination: Register, Memory Location

The source and the destination in an instruction cannot both be memory locations.

ADD AL, 74H; add the immediate number to 74H to the content of AL. Result in AL.

SUB: SUB Destination, Source.

These instructions subtract the number in some source from the number in some destination and put the result in the destination.

Source: Immediate Number, Register, or Memory Location.

Destination: Register or a Memory Location.

The source and the destination in an instruction cannot both be memory locations.

SUB AX, 3427H; Subtract immediate number 3427H from AX.

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.

MUL CX; Multiply AX with CX; result in high word in DX, low word in AX.

DIV: DIV Source.

This instruction is used to divide an unsigned word by a byte or to divide an unsigned double word (32 bits) by a word.

Source: Register, Memory Location.

If the divisor is 8-bit, then the dividend is in AX register. After division, the quotient is in AL and the remainder in AH.

If the divisor is 16-bit, then the dividend is in DX-AX register. After division, the quotient is in AX and the remainder in DX.

DIV CX; divide double word in DX and AX by word in CX; Quotient in AX; and remainder in DX.

Algorithm to add two 16-bit numbers

1. Load the first number in AX
2. Load the second number in BX

3 Add the second number to AX

4. Store the result in AX.

Algorithm to subtract two 16-bit numbers

1. Load the first number in AX.
2. Load the second number. in BX 3. Subtract the second number to AX

4. Store the result in AX.

Algorithm to multiply a 16-bit number by an 8-bit number

1. Load the first number in AX.
2. Load the second number. in BL
3. Multiply DX and AX.
4. The result is in DX and AX.

Algorithm to divide a 16-bit number by an 8-bit number

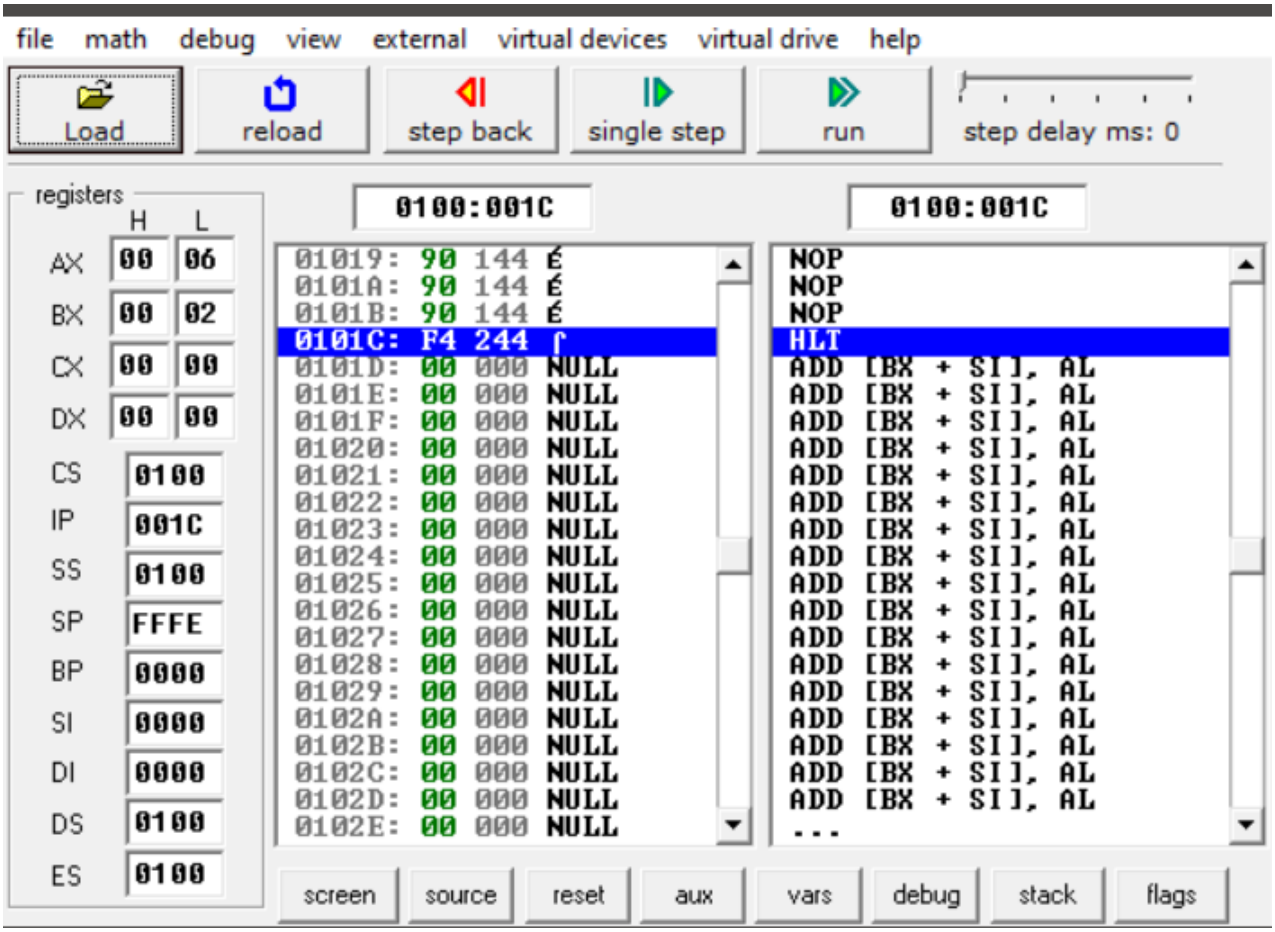
1. Load the first number in AX.
2. Load the second number. in BL
3. Divide AX by BL.
4. After division, the quotient is in AL and the remainder is in AH.

Program:

Addition of two 16 bit numbers:

```
MOV AX, 0004H
MOV BX, 0002H
ADD AX,BX
```

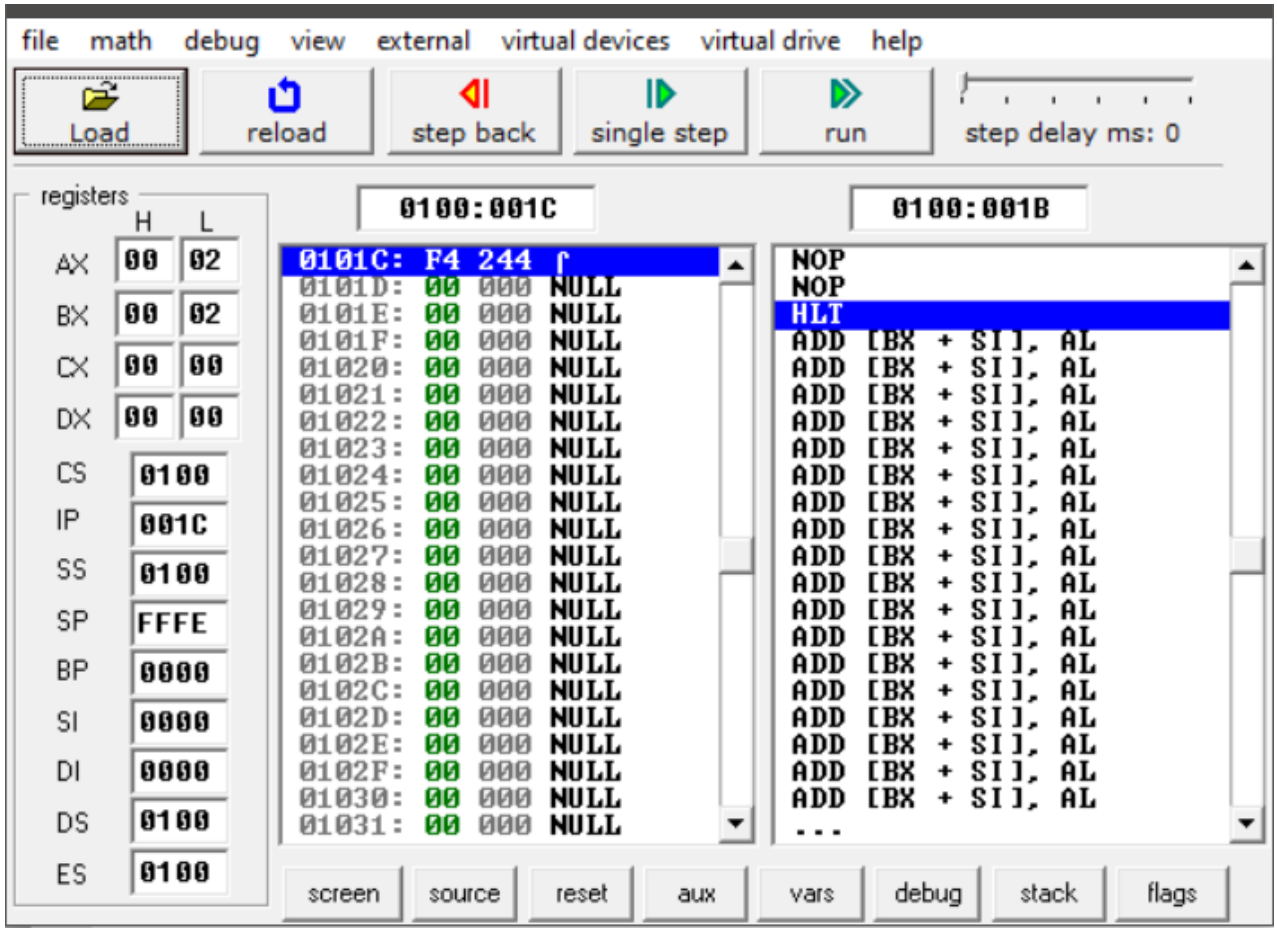
Output:-



Subtraction of two 16 bit numbers:

MOV AX, 0004H
MOV BX, 0002H
SUB AX,BX

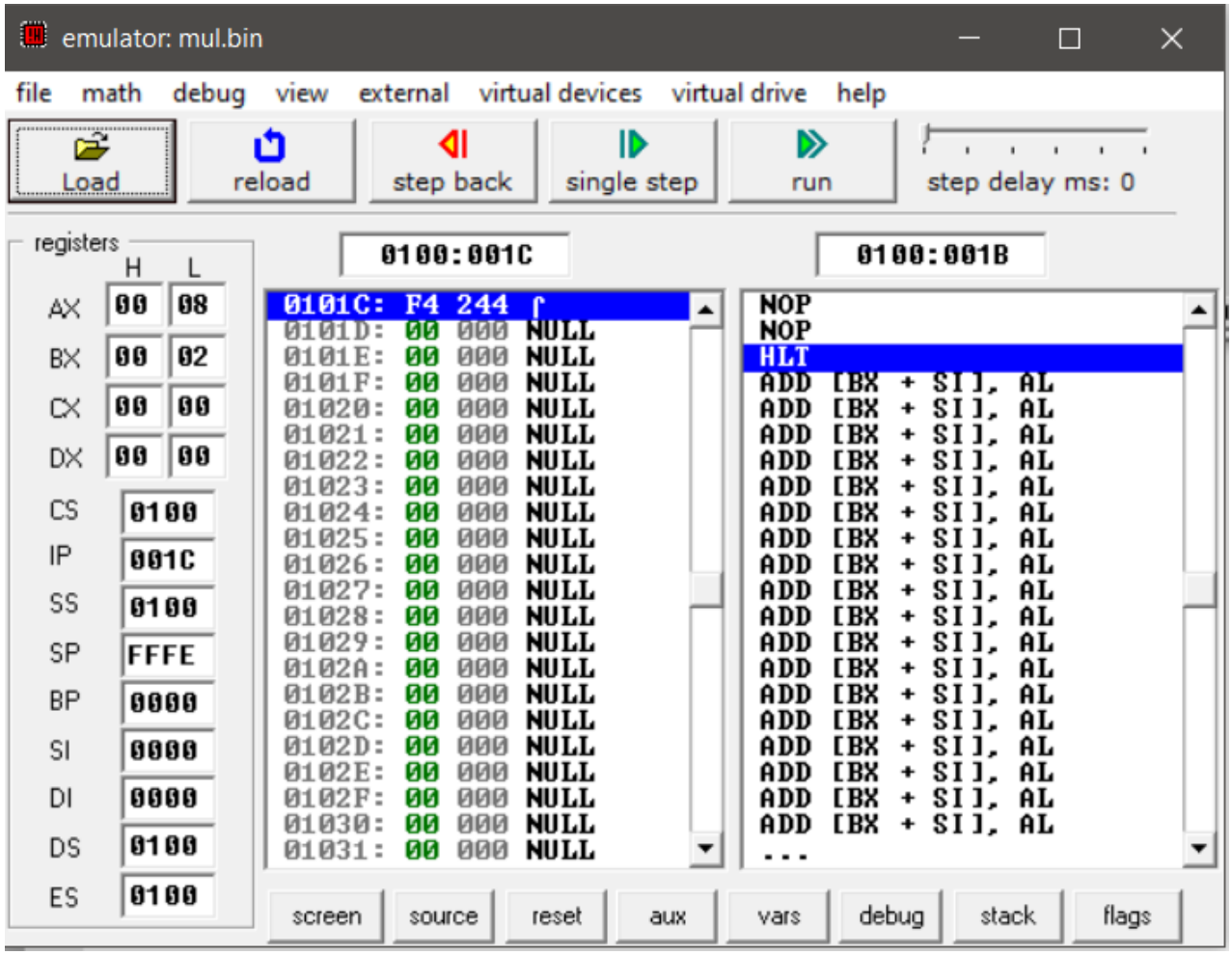
Output:



Multiplication of two 16 bit numbers:

```
MOV AX, 0004H
MOV BX, 0002H
MUL BX
```

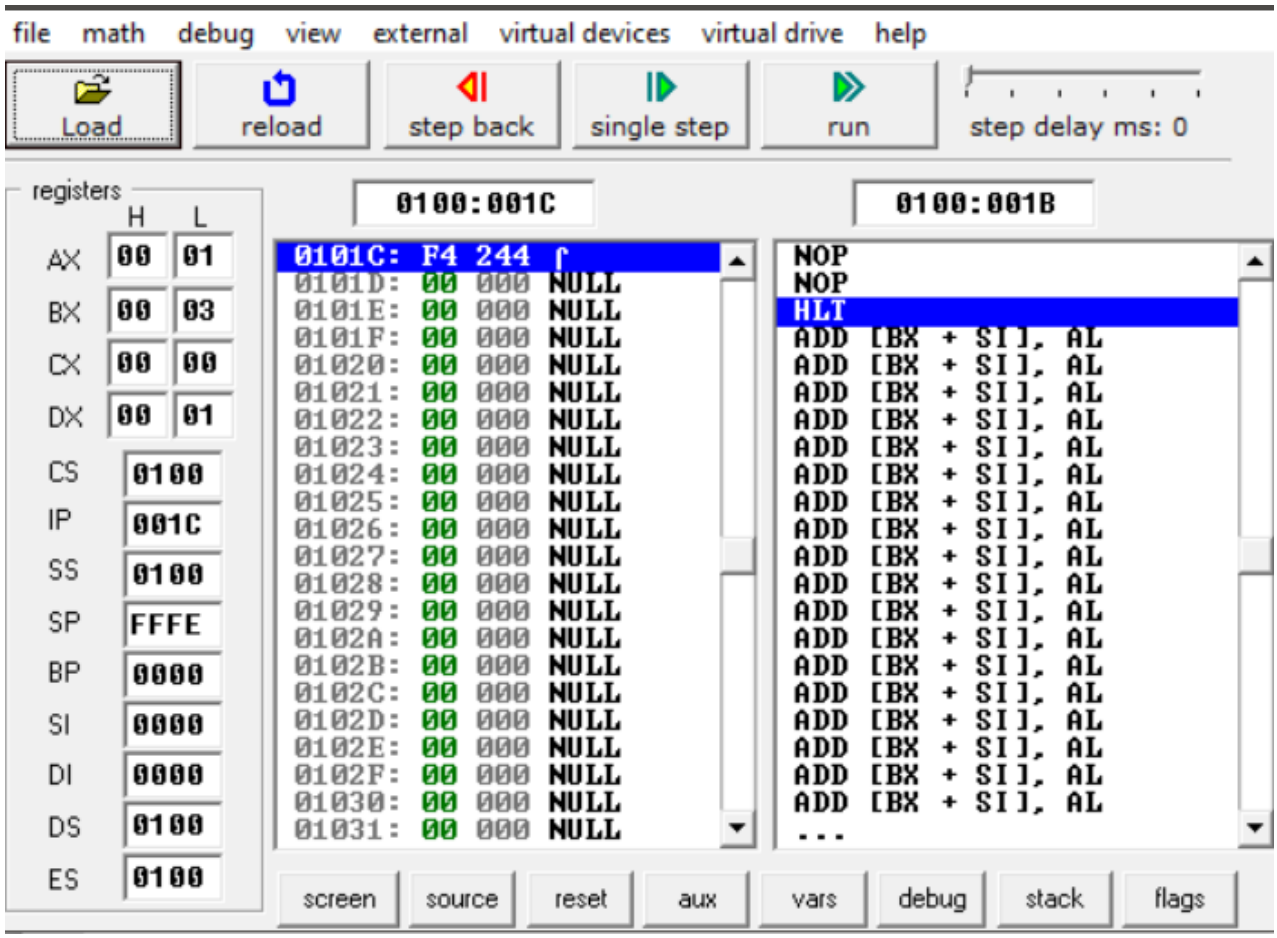
Output:



Division of two 16 bit numbers:

MOV AX, 0004H
MOV BX, 0003H
DIV BX

Output:



Conclusion: We successfully implemented basic arithmetic operations on 16-bit data in assembly language. The program showcased the fundamental arithmetic instructions available in assembly language and their usage for performing addition, subtraction, multiplication, and division operations.

1) Explain the features of 8086.

Ans. 16-bit Architecture: The 8086 is a 16-bit microprocessor, meaning it processes data and addresses in 16-bit chunks. It can access up to 1 MB of memory, providing a significant improvement over earlier 8-bit processors.

16-bit Registers: The 8086 includes several 16-bit registers, such as AX, BX, CX, DX, SI, DI, BP, and SP, which can store data and addresses. These registers are used for various purposes, including data manipulation, arithmetic operations, and memory addressing.

Segmented Memory Model: The 8086 employs a segmented memory model, dividing memory into segments of up to 64 KB each. It uses a combination of a 16-bit segment register and a 16-bit offset to address memory, allowing access to the entire 1 MB address space.

Instruction Set: The 8086 instruction set architecture (ISA) includes a wide range of instructions for data movement, arithmetic and logic operations, control flow, and string manipulation. Instructions are encoded using variable-length formats, with some instructions supporting multiple addressing modes.

Interrupts and Interrupt Handling: The 8086 supports both hardware and software interrupts, allowing external devices or software routines to interrupt normal program execution. It includes interrupt vectors and an interrupt vector table for handling different types of interrupts.

Pipelined Execution: The 8086 employs pipelined execution, allowing multiple instructions to be overlapped and executed concurrently to improve performance. Its pipeline includes instruction prefetch, instruction decoding, execution, and write-back stages.

Clock Speed: Early versions of the 8086 operated at clock speeds of around 5-10 MHz. Subsequent variants and improvements increased the clock speed, enhancing overall performance.

Compatibility and Legacy Support: The 8086 architecture and instruction set formed the basis for subsequent x86 processors, leading to a long legacy of compatibility and support for software written for earlier versions. Many modern x86 processors still maintain compatibility with 8086 instructions and programming models.

2) Explain general purpose and special purpose registers.

Ans. General-Purpose Registers: General-purpose registers are versatile storage locations that can hold various types of data and perform a wide range of operations. They are typically used for arithmetic and logical operations, memory addressing, data movement, and holding intermediate results during computation. The contents of general-purpose registers can be manipulated by instructions within the program. Examples of general-purpose registers include: AX, BX, CX, DX: These are 16-bit registers (also accessible as 8-bit parts, e.g., AL, AH) used for general arithmetic and data manipulation. SI, DI: Source Index and Destination Index registers, commonly used for string operations and memory copying. BP, SP: Base Pointer and Stack Pointer registers, used for addressing data on the stack and

accessing parameters and local variables in procedures. CS, DS, SS, ES: Code Segment, Data Segment, Stack Segment, and Extra Segment registers, used for memory segmentation in the x86 architecture.

Special-Purpose Registers: Special-purpose registers have specific predefined functions and are typically used for controlling the operation of the processor or storing critical system state information. They may not be directly accessible or modifiable by the instructions within the program and are often managed by the processor hardware. Special-purpose registers play a crucial role in controlling program execution, managing interrupts, handling exceptions, and facilitating communication between hardware components. Examples of special-purpose registers include: Instruction Pointer (IP or EIP): Holds the memory address of the next instruction to be executed. Flags Register (FLAGS or EFLAGS): Stores various status flags indicating the outcome of arithmetic/logical operations, control flow conditions, and processor state information. Control Registers (CR0, CR2, CR3, CR4): Used for controlling processor operation and managing memory management features like paging and virtual memory. Debug Registers (DR0-DR7): Used for hardware-assisted debugging and monitoring of program execution.