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.:	**1**
Title of the Experiment:	**Program to perform basic arithmetic operations on 16 bit data**
Date of Performance:	**16/01/2025**
Date of Submission:	**23/01/2025**

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: 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.

Code:

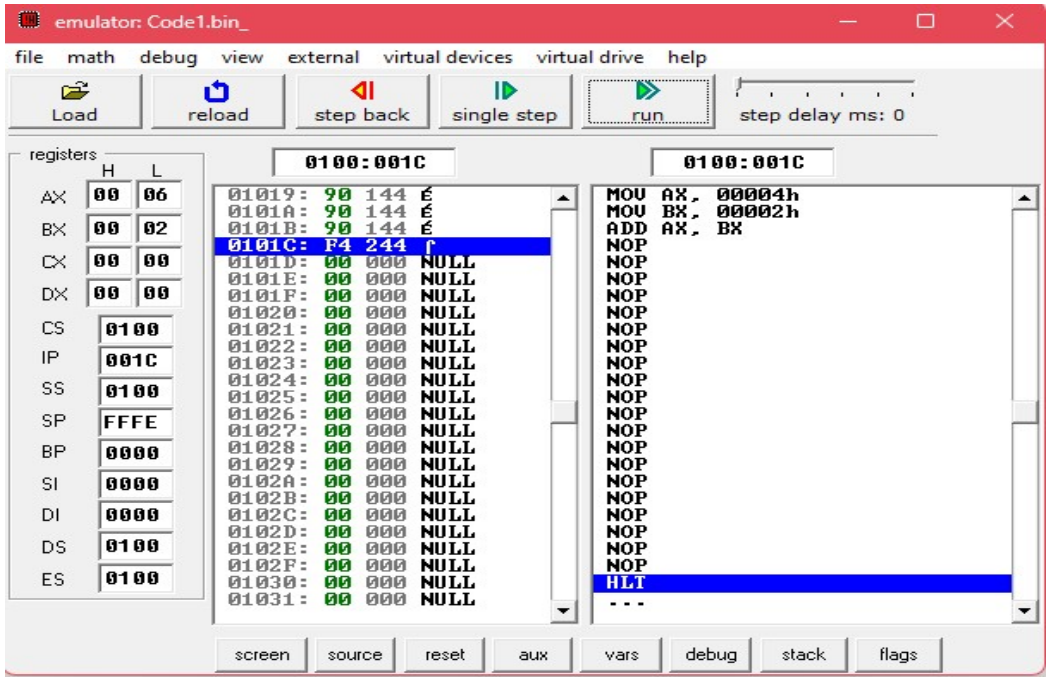
```
MOV AX,0004H
MOV BX,0002H
ADD AX, BX
```

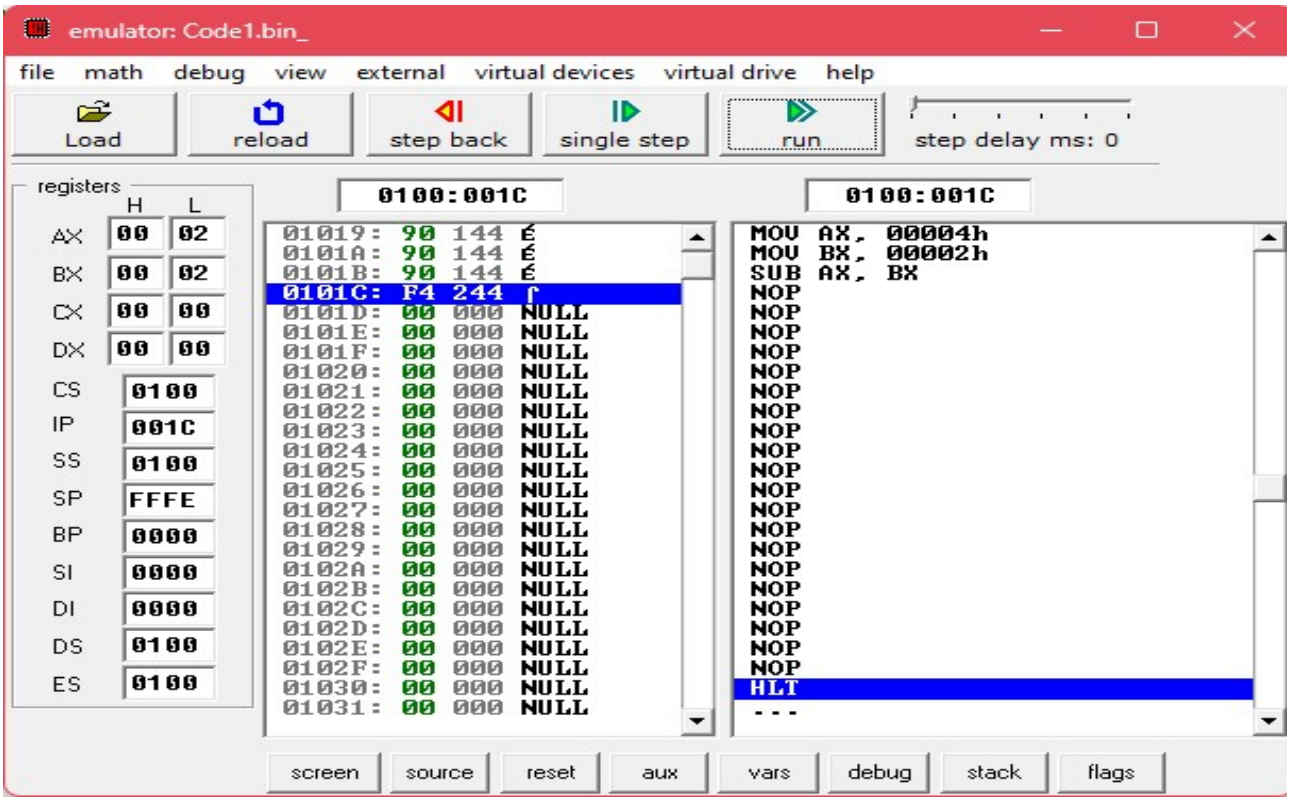
```
MOV AX,0004H
MOV BX,0002H
SUB AX, BX
```

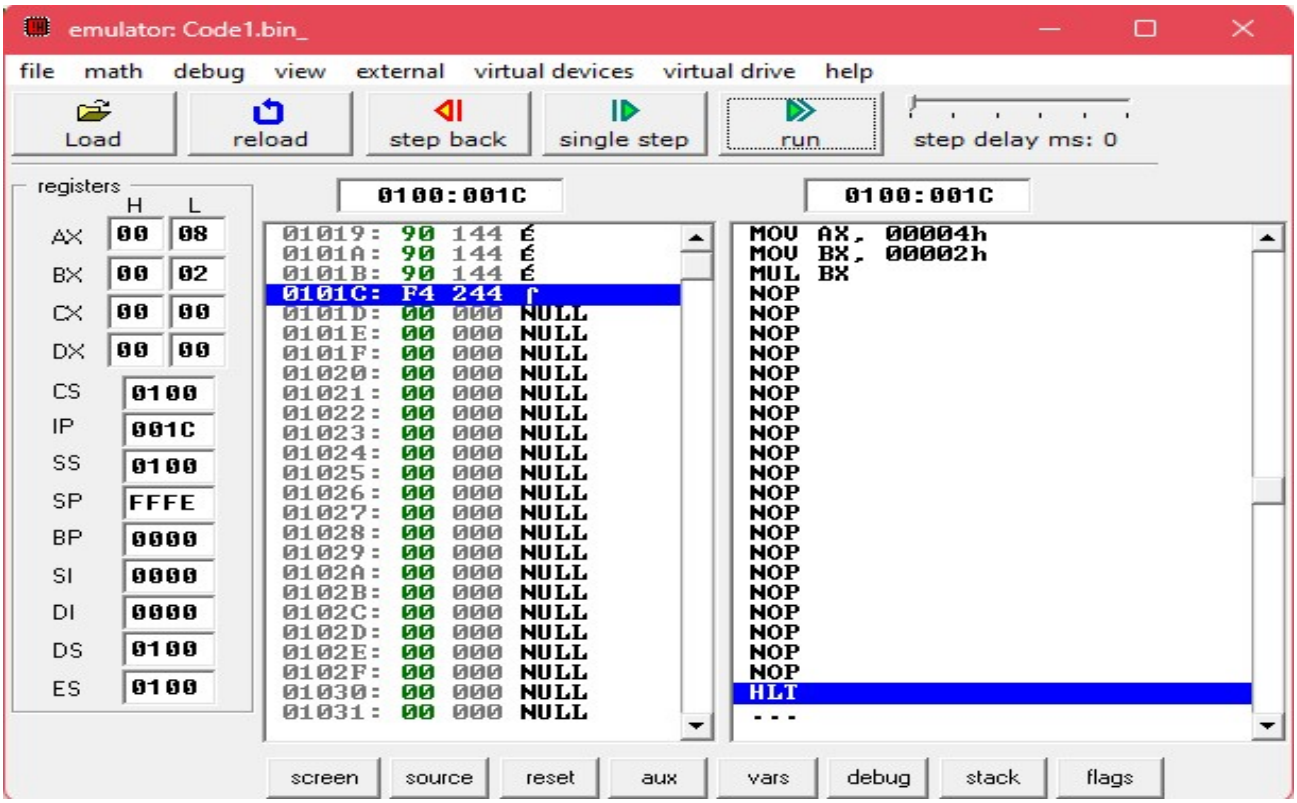
```
MOV AX,0004H
MOV BX,0002H
MUL BX
```

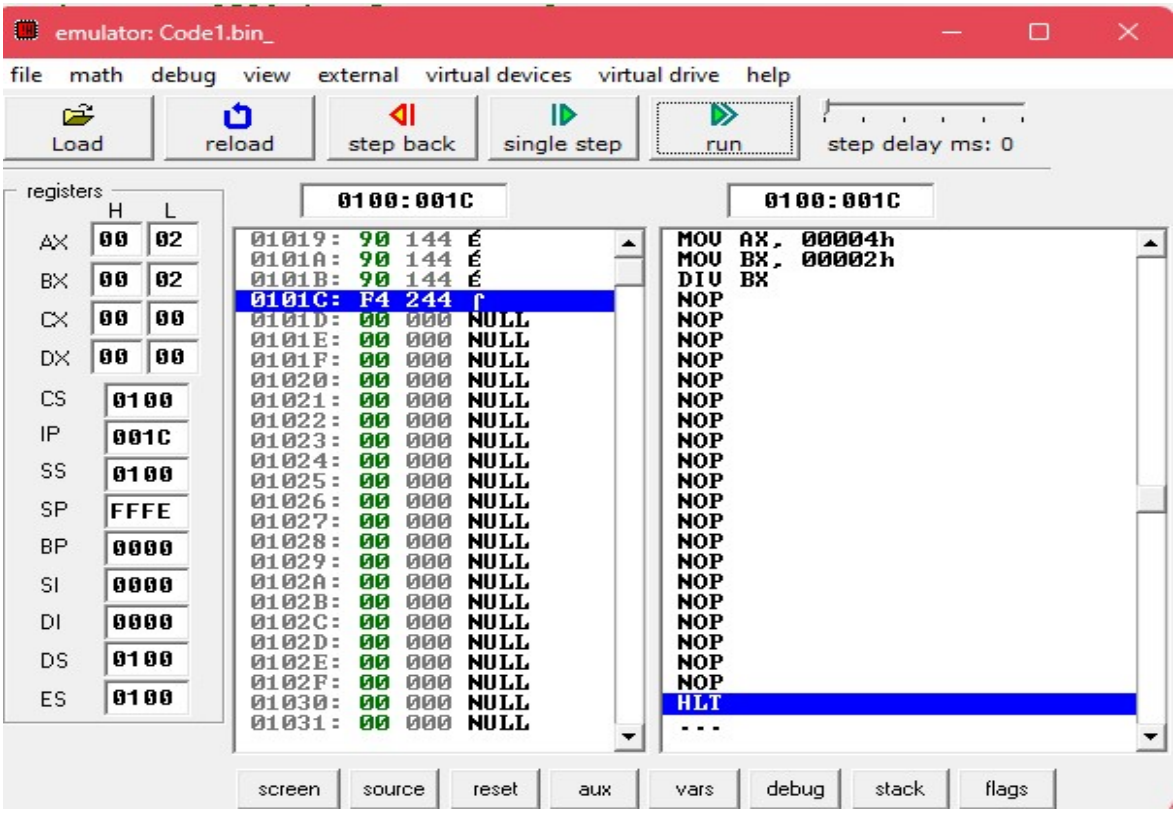
```
MOV AX,0004H
MOV BX,0002H
DIV BX
```

Output:




emulator: Code1.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 00 02
BX 00 02
CX 00 00
DX 00 00
CS 0100
IP 001C
SS 0100
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0100
ES 0100
0100:001C
0100:001C
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: F4 244 ⌠
0101D: 00 000 NULL
0101E: 00 000 NULL
0101F: 00 000 NULL
01020: 00 000 NULL
01021: 00 000 NULL
01022: 00 000 NULL
01023: 00 000 NULL
01024: 00 000 NULL
01025: 00 000 NULL
01026: 00 000 NULL
01027: 00 000 NULL
01028: 00 000 NULL
01029: 00 000 NULL
0102A: 00 000 NULL
0102B: 00 000 NULL
0102C: 00 000 NULL
0102D: 00 000 NULL
0102E: 00 000 NULL
0102F: 00 000 NULL
01030: 00 000 NULL
01031: 00 000 NULL
MOV AX, 00004h
MOV BX, 00002h
SUB AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
---
screen source reset aux vars debug stack flags


emulator: Code1.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 00 08
BX 00 02
CX 00 00
DX 00 00
CS 0100
IP 001C
SS 0100
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0100
ES 0100
0100:001C
0100:001C
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: F4 244 ⌠
0101D: 00 000 NULL
0101E: 00 000 NULL
0101F: 00 000 NULL
01020: 00 000 NULL
01021: 00 000 NULL
01022: 00 000 NULL
01023: 00 000 NULL
01024: 00 000 NULL
01025: 00 000 NULL
01026: 00 000 NULL
01027: 00 000 NULL
01028: 00 000 NULL
01029: 00 000 NULL
0102A: 00 000 NULL
0102B: 00 000 NULL
0102C: 00 000 NULL
0102D: 00 000 NULL
0102E: 00 000 NULL
0102F: 00 000 NULL
01030: 00 000 NULL
01031: 00 000 NULL
MOV AX, 00004h
MOV BX, 00002h
MUL BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
---
screen source reset aux vars debug stack flags



Conclusion:

The assembly code provided performs basic arithmetic operations on 16-bit data by utilizing the x86 assembly language instructions for addition, subtraction, multiplication, and division. These operations demonstrate the fundamental principles of manipulating data at the machine level.

1. Explain the features of 8086.

Features of 8086:

- **16-bit Processor:** It processes 16 bits of data at a time.
- **16-bit Address Bus:** Supports addressing up to 1MB of memory.
- **Segmented Memory Architecture:** Divides memory into segments (Code, Data, Stack, and Extra).
- **Two Operating Modes:** Minimum mode (single processor) and Maximum mode (multiple processors).
- **Instruction Set:** Supports a rich set of instructions for arithmetic, logical, and data movement operations.
- **Pipelining:** The 8086 uses pipelining to fetch and decode instructions while executing previous ones, increasing efficiency.
- **Clock Speed:** Can operate at clock speeds from 5 to 10 MHz.
- **Registers:** It has a variety of registers for different operations, such as AX, BX, CX, DX, and more.

2. Explain general purpose and special purpose registers.

General Purpose and Special Purpose Registers:

General Purpose Registers (GPRs):

- **AX (Accumulator):** Used for arithmetic and data transfer operations.
- **BX (Base):** Used for addressing, often holds the base address.
- **CX (Count):** Used for loops and string operations, often serves as a counter.
- **DX (Data):** Used for I/O operations, holds data for multiplication and division.

Special Purpose Registers:

-
- **IP (Instruction Pointer):** Holds the address of the next instruction to be executed.
- **SP (Stack Pointer):** Points to the top of the stack in memory.
- **BP (Base Pointer):** Points to the base of the current stack frame.
- **SI (Source Index):** Used for string operations, points to source data.
- **DI (Destination Index):** Used for string operations, points to destination data.
- **Flags Register (PF, CF, ZF, SF, OF, etc.):** Holds status flags for operation results (carry, zero, sign, etc.).