AY: 2024-25

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

Name of Student:	Bhagyashri Kaleni Sutar
Roll No. :	**75**
Experiment No.:	**1**
Title of the Experiment:	**Program to perform basic arithmetic operations on 16 bit data**
Date of Performance:	
Date of Submission:	

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)	Meet Expect 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 :

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.

<u>Algorithm to add two 16-bit numbers</u>

- Load the first number in AX
- 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

- Load the first number in AX.
- Load the second number. in BX
- Subtract the second number to AX
- Store the result in AX.

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

- Load the first number in AX.
- Load the second number. in BL
- Multiply DX and AX.
- The result is in DX and AX.

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

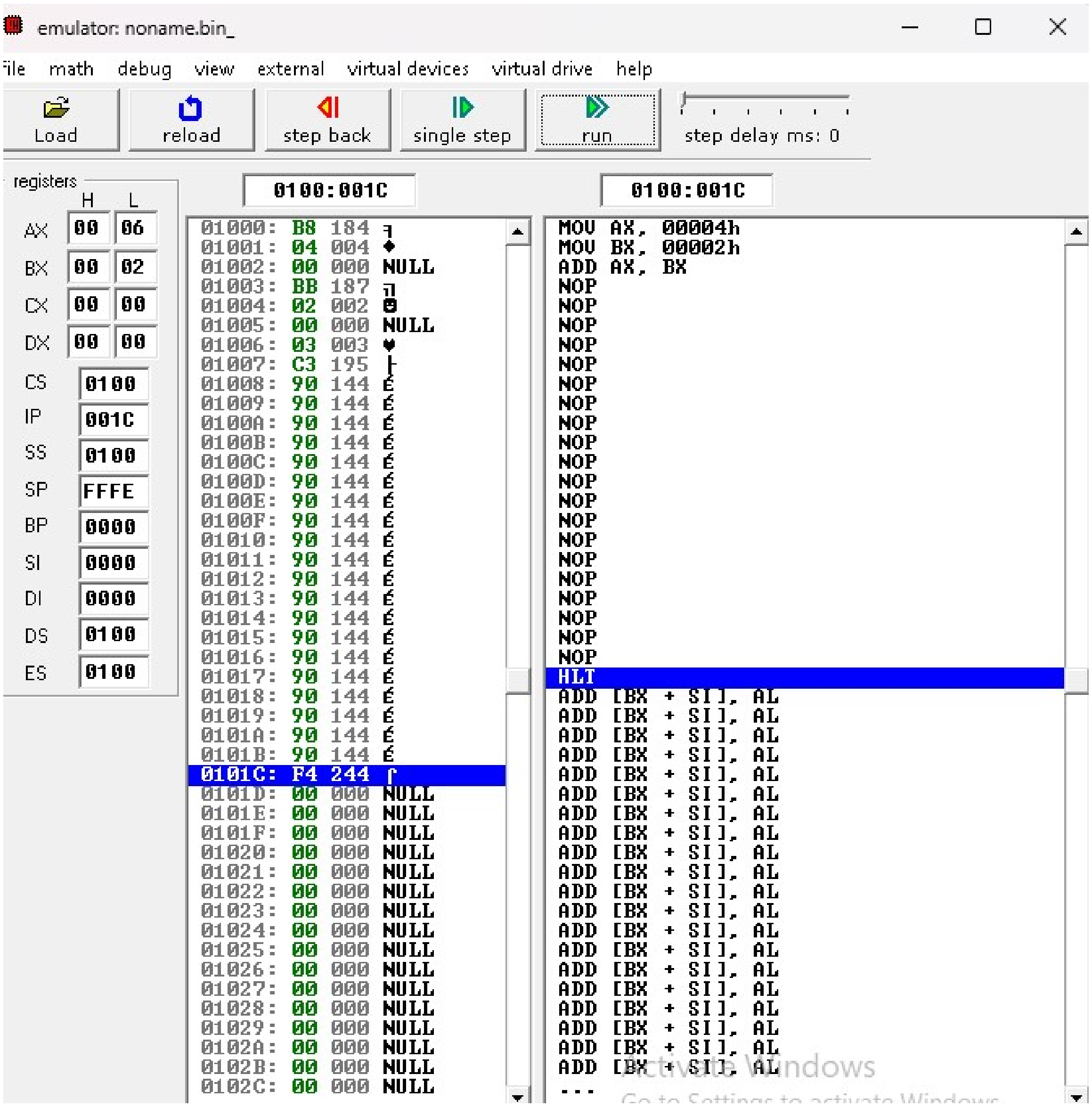
- Load the first number in AX.
- Load the second number. in BL
- Divide AX by BL.
- After division, the quotient is in AL and the remainder is in AH.

Code:

Addition:

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

Output:-



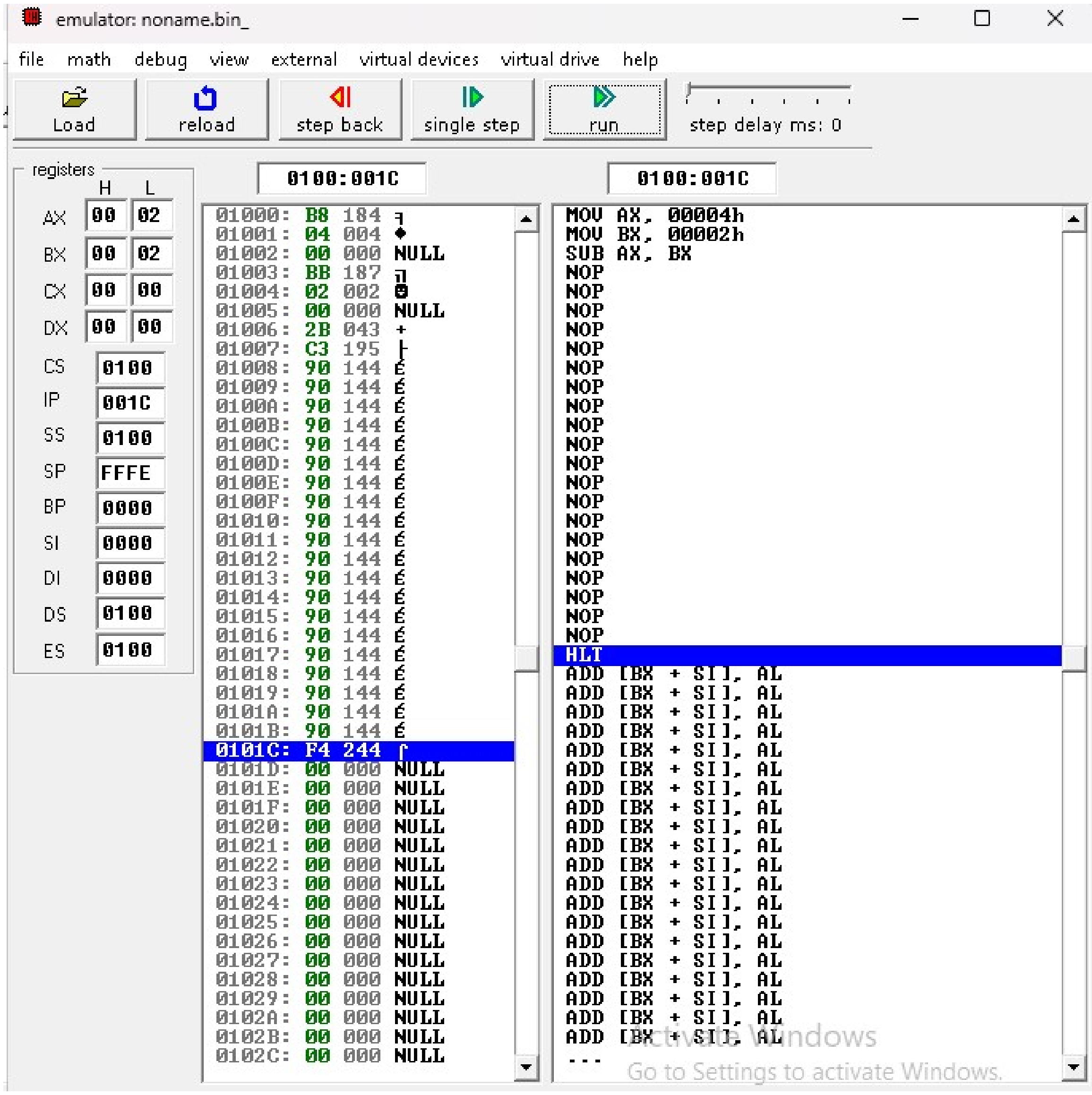
Subtraction:-

MOV AX,0004H

MOV BX,0002H

SUB AX, BX

Output:-



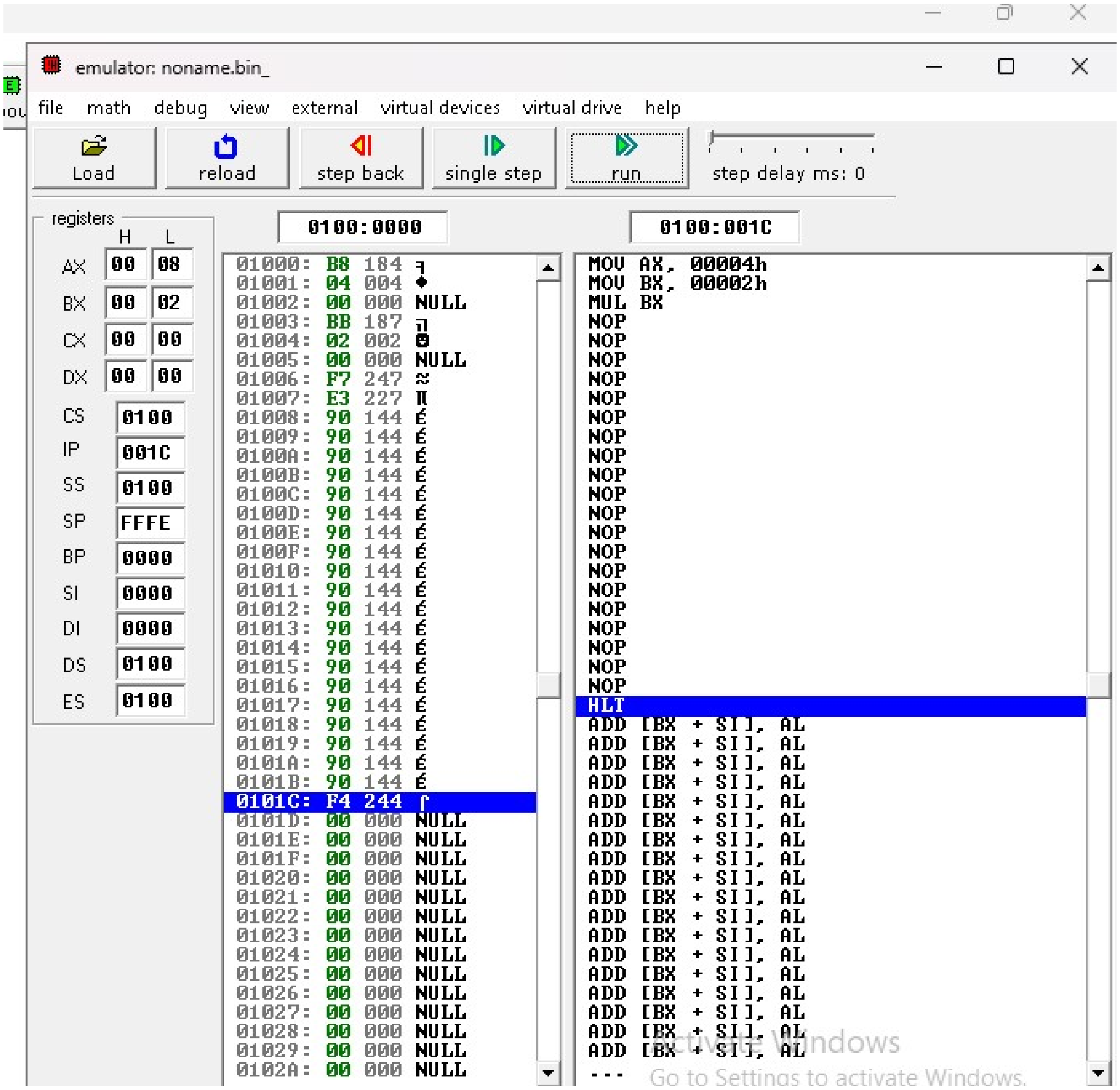
Multiplication:-

MOV AX,0004H

MOV BX,0002H

MUL BX

Output:-



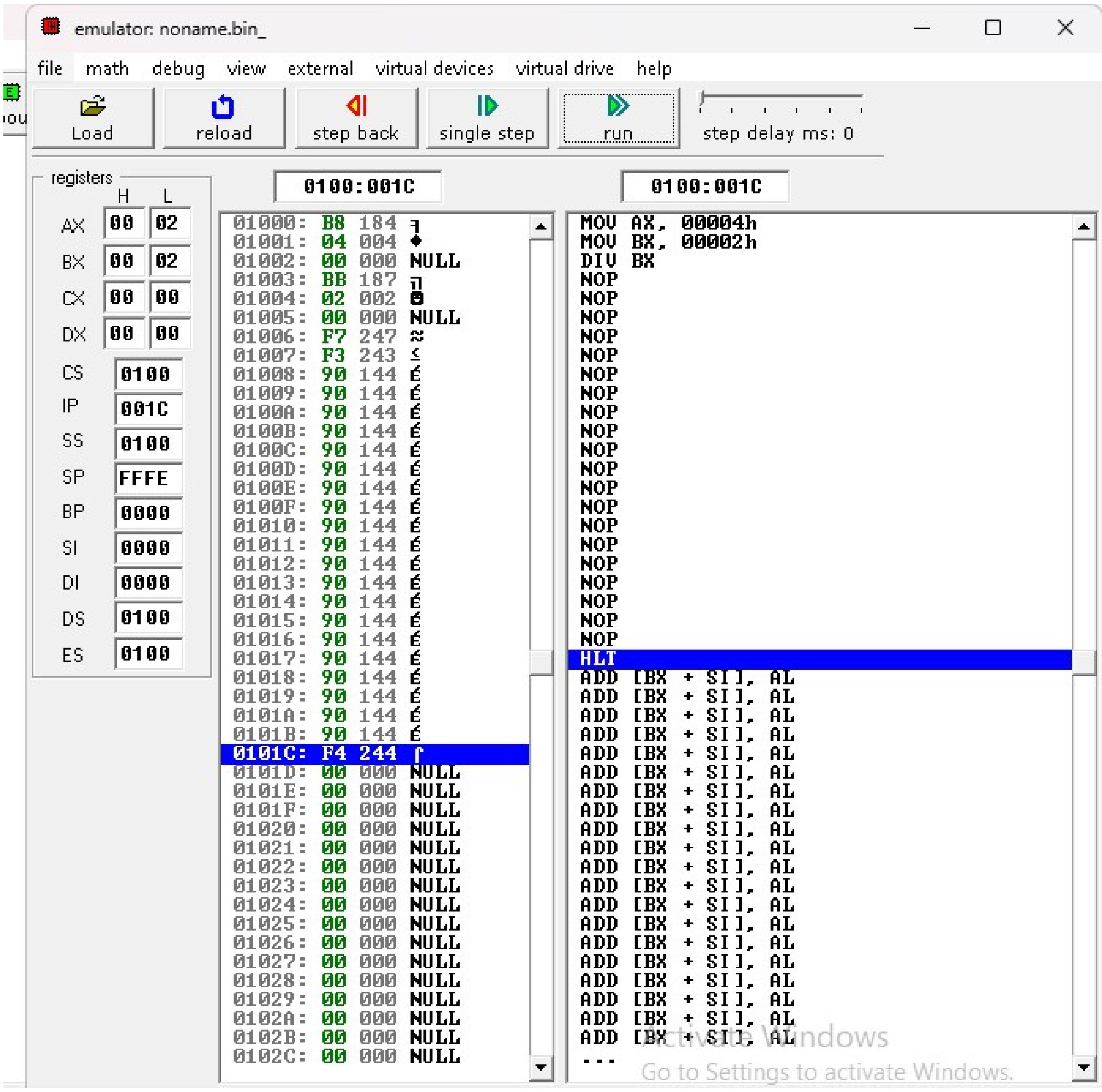
Division:-

MOV AX,0004H

MOV BX,0002H

DIV BX

Output:-



Conclusion:

- Explain the features of 8086.

Ans:-16-bit Microprocessor: The 8086 processes data 16 bits at a time, which was advanced for its time.

16-bit Data Bus: It has a 16-bit wide data bus, allowing it to transfer data more efficiently than previous processors.

20-bit Address Bus: The 8086 has a 20-bit address bus, which allows it to address up to 1 MB (2^20) of memory. This was a significant advancement over previous processors with smaller address buses.

Registers:

4 general-purpose 16-bit registers: AX, BX, CX, DX.
4 segment registers: CS (Code Segment), DS (Data Segment), SS (Stack Segment), ES (Extra Segment).
2 index registers: SI (Source Index) and DI (Destination Index).
A 16-bit flag register.
Instruction Set: The 8086 has a rich set of instructions that include arithmetic, logical, control, and data transfer operations. It supports both 8-bit and 16-bit operations.

Segmented Memory: The 8086 uses a segmented memory model, dividing memory into segments. The 20-bit address allows for 64 KB segments, enabling access to the full 1 MB address space.

Two Operating Modes:

Minimum Mode: Used when there is no external controller, and the processor controls everything.
Maximum Mode: Used with external controllers for more complex systems (e.g., multi-processor systems).
Clock Speed: The 8086 typically operates at clock speeds ranging from 5 MHz to 10 MHz, with the higher clock speeds providing better performance.

Multiplexed Address/Data Bus: The 8086 has a multiplexed bus, meaning the address and data lines share the same physical lines, but they are used at different times to maximize efficiency.

Pipelined Architecture: The 8086 has a 6-byte instruction queue, which helps pre-fetch instructions and allows the CPU to fetch the next instruction while executing the current one, improving performance.

- Explain general purpose and special purpose registers.

Ans:-1. General Purpose Registers:
These are used to store temporary data during computation. General-purpose registers are flexible and can be used for a variety of operations, such as arithmetic calculations, data movement, or holding intermediate values.

In the 8086 microprocessor, there are 4 general-purpose registers:
AX (Accumulator Register):

Used in arithmetic, logical, and data transfer operations.
It is often used as the primary register for operations (e.g., for multiplication, division, I/O operations).
Divided into two 8-bit registers: AH (high byte) and AL (low byte).
BX (Base Register):

Used for base addressing in memory.
Often used to hold a pointer to data in the data segment.
CX (Count Register):

Used for loop counters and operations that require counting (like in shift operations or repeated string operations).
Divided into CH (high byte) and CL (low byte).
DX (Data Register):

Used for I/O operations and data handling.
It also plays a role in multiplication and division (the result of a 32-bit multiply operation is stored in DX:AX).
These general-purpose registers can be used interchangeably in most operations, and they support 16-bit operations. Their names are mostly historical or have been adopted based on how they were used in the original architecture.

2. Special Purpose Registers:
These registers have specific, dedicated functions and are essential for controlling various aspects of the processor's operation, such as memory segmentation, program flow, and system status. In the 8086 processor, there are several special-purpose registers:

CS (Code Segment Register):

Holds the segment address of the code, which is the part of the memory that contains the executable instructions.
Works with the instruction pointer (IP) to point to the next instruction to be executed.
DS (Data Segment Register):

Holds the segment address for data storage.
Used by the processor to access the data area in memory.
SS (Stack Segment Register):

Holds the segment address of the stack, which is used to store temporary data such as function calls, return addresses, and local variables.
Works with the stack pointer (SP) to manage the stack.
ES (Extra Segment Register):

Used for additional data storage, often when working with string operations or during memory-mapped I/O.
IP (Instruction Pointer):

Holds the offset address of the next instruction to be executed within the code segment.
It is automatically updated as instructions are executed.
SP (Stack Pointer):

Holds the offset address of the top of the stack in the stack segment.
It points to the current position in the stack, which is used to push or pop values.
BP (Base Pointer):

Points to the base of the current stack frame. It is used for referencing function parameters and local variables in stack-based operations.
SI (Source Index Register):

Primarily used for string and memory operations.
It can be used to point to the source data in memory when performing string manipulations or data transfer.
DI (Destination Index Register):

Used for string operations and points to the destination data location in memory.
Flags Register:

It is a 16-bit register that holds status flags and control flags used for different operations and decisions within the CPU.
Some of the important flags are:
Carry Flag (CF): Indicates carry or borrow during arithmetic operations.
Zero Flag (ZF): Set if the result of an operation is zero.
Sign Flag (SF): Set if the result is negative.
Overflow Flag (OF): Set if an overflow occurs in signed arithmetic operations.