

8-Bit Arithmetic Operations using 8085

Aim:

To perform 8-bit arithmetic operations such as addition, subtraction, multiplication, and division using the 8085 microprocessor.

Apparatus Required:

- Laptop with internet connection

Algorithm:

For Addition (With Carry Consideration):

- Load the first number into register A.
- Load the second number into register B.
- Add the contents of registers A and B.
- If carry is generated, store carry in a separate location.
- Store the sum in another location.

For Subtraction (Considering Greater Number):

- Load the first number into register A.
- Load the second number into register B.
- Compare A and B.
- If $A < B$, swap the values of A and B to ensure positive result.
- Subtract the content of B from A.
- Store the result in a specified location.

For Multiplication:

- Load the first number into register A.
- Load the second number into register B.
- Multiply A and B using repeated addition.
- Store the result in suitable locations (including extra space if needed for higher bits).

For Division:

- Load the dividend into register A.
- Load the divisor into register B.
- Perform division using repeated subtraction.

- Store the quotient in one location and remainder in another.

Program:

Addition of Two 8-bit Numbers:

```
IN 01H      ; Read first number into A
MOV B, A     ; Store it in B
IN 02H      ; Read second number into A
ADD B        ; A = A + B
OUT 03H      ; Output sum to port 03H
```

```
MVI C, 00H   ; Clear C register
JNC SKIP_CARRY ; Jump if no carry
INR C         ; If carry occurred, C = 1
```

```
SKIP_CARRY:
MOV A, C
OUT 04H      ; Output carry to port 04H
```

Subtraction (First number - Second number)

```
IN 01H      ; Read first number into A
MOV B, A     ; Store in B
IN 02H      ; Read second number into A
MOV C, A     ; Store in C
MOV A, B     ; A = first number
SUB C        ; A = A - second number
OUT 05H      ; Output result to port 05H
```

```
HLT          ; End of program
```

Multiplication using repeated addition:

```
IN 01H      ; Read first number (Multiplicand) into A
MOV C, A     ; Store in C
```

```
IN 02H      ; Read second number (Multiplier) into A
MOV B, A     ; Store in B
```

```
MVI A, 00H   ; Clear A to hold result
```

```
LOOP:
ADD C        ; A = A + C
DCR B        ; B = B - 1
JNZ LOOP     ; Repeat until B = 0
```

```
OUT 06H      ; Output the result to port 06H
HLT          ; End of program
```

Division (Using Repeated Subtraction):

```
IN 01H      ; Read dividend into A
MOV C, A     ; Store dividend in C (for remainder tracking)
MVI A, 00H   ; Clear A for quotient
MOV D, A     ; Use D to store quotient
```

```
IN 02H      ; Read divisor into A
MOV B, A     ; Store divisor in B
```

```
DIV_LOOP:
MOV A, C     ; Load current remainder into A
CMP B        ; Compare remainder with divisor
JC END_DIV   ; If A < B, jump to END_DIV
SUB B        ; A = A - B
MOV C, A     ; Update remainder in C
INR D        ; Increment quotient
JMP DIV_LOOP ; Repeat loop
```

```
END_DIV:
MOV A, D     ; Move quotient to A
OUT 03H      ; Output quotient to port 03H
```

```
MOV A, C     ; Move remainder to A
OUT 04H      ; Output remainder to port 04H
```

```
HLT          ; End program
```

Output:

Addition of Two 8-bit Numbers:

Input Ports:

- **01H** → First number
- **02H** → Second number

Output Ports:

- **03H** → Sum
- **04H** → Carry (if generated)

The screenshot shows an 8086 assembly editor interface. On the left, there's a panel for I/O Ports with a list of ports from 0x00 to 0x1A. The main area displays assembly code for adding two 8-bit numbers. The code reads two numbers from ports 01H and 02H, adds them, and outputs the sum to port 03H and the carry to port 04H. On the right, a 'Machine Code' panel shows the corresponding machine code for each instruction.

```
1 IN 01H ; Read first number into A
2 MOV B, A ; Store it in B
3 IN 02H ; Read second number into A
4 ADD B ; A = A + B
5 OUT 03H ; Output sum to port 03H
6
7 MVI C, 00H ; Clear C register
8 JNC SKIP_CARRY ; Jump if no carry
9 INR C ; If carry occurred, C = 1
10
11 SKIP_CARRY:
12 MOV A, C
13 OUT 04H ; Output carry to port 04H
14
15 HLT ; End of program
```

Line	Address	Machine Code	Source Code
1	0x0	D8 01	IN 01H ; Read first number into A
2	0x2	47	MOV B, A ; Store it in B
3	0x3	D8 02	IN 02H ; Read second number into A
4	0x5	80	ADD B ; A = A + B
5	0x6	D3 03	OUT 03H ; Output sum to port 03H
6			
7	0x8	0E 00	MVI C, 00H ; Clear C register
8	0xA	D2 0E 00	JNC SKIP_CARRY ; Jump if no carry
9	0xD	0C	INR C ; If carry occurred, C = 1
10			
11			SKIP_CARRY:
12	0xE	79	MOV A, C
13	0xF	D3 04	OUT 04H ; Output carry to port 04H
14			
15	0x11	76	HLT ; End of program

Subtraction (First number - Second number)

Input Ports:

- **01H** → First number
- **02H** → Second number

Output Ports:

- **05H** → Result (Difference)

The screenshot shows an assembly editor with the following components:

- I/O Ports:** A list of ports from 0x00 to 0x0F. Port 0x01 is highlighted with a value of 60, and port 0x05 is highlighted with a value of 30.
- Assembly Code:**

```

1 IN 01H      ; Read first number into A
2 MOV B, A    ; Store in B
3 IN 02H      ; Read second number into A
4 MOV C, A    ; Store in C
5 MOV A, B    ; A = first number
6 SUB C       ; A = A - second number
7 OUT 05H     ; Output result to port 05H
8
9 HLT         ; End of program

```
- Machine Code:** A table showing the compiled machine code for each instruction.

Line	Address	Machine Code	Source Code
1	0x0	0B 01	IN 01H ; Read first number into A
2	0x2	47	MOV B, A ; Store in B
3	0x3	0B 02	IN 02H ; Read second number into A
4	0x5	4F	MOV C, A ; Store in C
5	0x6	78	MOV A, B ; A = first number
6	0x7	91	SUB C ; A = A - second number
7	0x8	D3 05	OUT 05H ; Output result to port 05H
8			
9	0xA	76	HLT ; End of program

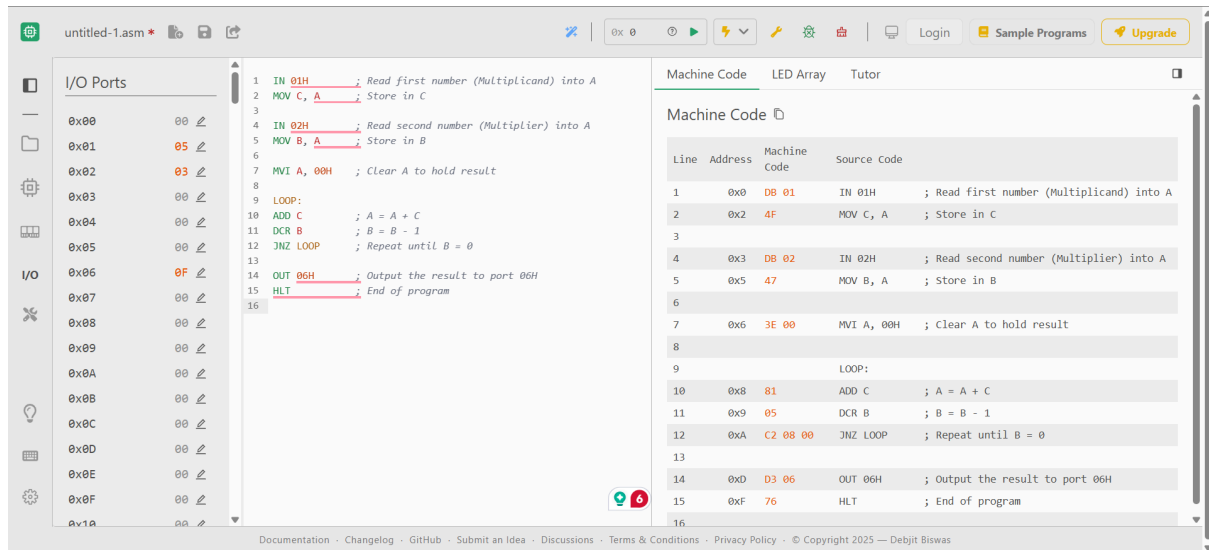
Multiplication using repeated addition:

Input Ports:

- **01H** → Multiplicand
- **02H** → Multiplier

Output Ports:

- **06H** → Product



Division (Using Repeated Subtraction):

Input Ports:

- **01H** → Dividend
- **02H** → Divisor

Output Ports:

- **03H** → Quotient
- **04H** → Remainder

The 8-bit arithmetic operations using the 8085 microprocessor have been successfully executed and verified using memory access for input and output.