

// Addition - UN/SIGNED MANTISSA AND UN/SIGNED EXPONENT

```
START:  LXI SP,4000
        LXI H,2000
        MOV B,M    // Saving first exponent in B
        INX H
        PUSH H
        MOV A,M    // Saving Second Exponent in A
        CMP B      // Comparing both the exponents
        JZ CONTINUE // If both the exponents ->Performing Add
// IF both exponents are not equal --> subtracting
        MOV C,B    // Transferring A->B and B->A
        MOV B,A
        MOV A,C
        ADI 00
        JM 1COMP
        JP NEXTEXPO

1COMP:   CMA
        ADI 01
        ADI F8
        MOV C,A

NEXTEXPO:  MOV A,B
        ADI 00
        JP SUBTRACT
        ANI 07
        ADD C
        JMP GO

SUBTRACT:  MOV A,C    // Moving the number in A for Subtraction
```

SUB B // Getting the difference

```
GO:    MOV B,A    // Transferring the result to B
      INX H // HL = 2003H
      PUSH H
      MOV A,M    // M has first number
      ADI 00     // Adding 00 to first number to check sign bit
      JM NEWROTATE // IF Sign bit=1 then --> NEWROTATE

LOOP:  RAL  // Multiplying 2
      DCR B // Decrementing B
      JNZ LOOP // If B!=0 --> Repeat
      MOV M,A // If B!=0 then Moving the new rotated number to M
      JZ NEXT  // If B==0 --> Jump to NEXT

NEWROTATE: ADI 80 // Making Sign Bit 0
      CMC

LOOP2:  RAL  // Multiplying by 2
      DCR B
      JNZ LOOP2
      ADI 80
      MOV M,A
      JMP NEXT

CONTINUE: INX H
      PUSH H
      MOV A,M

NEXT:   ADI 00 // For 2's complement if number is -ve or positive
      JM 1TWO  // IF number is negative --> Jumping to 1TWO
```

JP SECONDNUMBER

1TWO: CMA
ADI 01
ADI 80
MOV M,A

SECONDNUMBER: MOV B,A // Moving first number to B
INX H
PUSH H
MOV A,M
ADI 00 // For 2's complement if number is -ve or positive
JM 2TWO // IF number is negative --> Jumping to 1TWO
JP MOVE

2TWO: CMA
ADI 01
ADI 80
MOV M,A

MOVE: MOV C,A // Moving Second Number to C

NOWAD: POP H// HL = 2002
MOV A,M // MOVING FIRST Number to M
STC
CMC
RAL // For Checking Overflow
MOV D,A
POP H// HL = 2003
MOV A,M
STC

```
CMC
RAL    // For Checking
ADD D    // Adding for getting carry from 7th bit
JNC NEXT3
MVI E,01
```

```
NEXT3: MOV A,C    // Adding the unrotated numbers
ADD B// Adding with B
MOV D,A
JC CHECKINGOVERFLOW    // If Carry is Generated --> JMP
MVI A,00
CMP E
JZ OLDADD
JNZ NEWADD
```

```
CHECKINGOVERFLOW: MVI A,01
CMP E
JZ OLDADD
JNZ NEWADD
```

```
OLDADD:    MOV B,D
MOV A,B    // To do 2's complement of the answer
ADI 00    // To check sign bit
JM SIGNEDBIT// If S = 1 --> SIGNEDBIT
JP EXPONENTS
HLT
```

```
NEWADD:    MVI E,01
MOV A,B    // For New Addition Dividing by 2 both the numbers
ADI 00    // Two Check Sign bit
JM YES1    // If Sign Bit is 1 then --> YES1
```

RAR // If S = 0 then --> Simply rotating

MOV B,A

JMP NEXTNUMBER

YES1: ADI 80 // To remove Sign bit

CMC

RAR

ADI 80 // To add Sign bit

MOV B,A

NEXTNUMBER: MOV A,C

ADI 00 // Two Check Sign bit

JM YES2 // If Sign Bit is 1 then --> YES1

RAR // If S = 0 then --> Simply rotating

MOV C,A

JMP FINALANSWEROFROTATED

YES2: ADI 80 // To remove Sign bit

CMC

RAR

ADI 80 // To add sign bit

MOV C,A

FINALANSWEROFROTATED: ADD B

MOV B,A

MOV A,B // To do 2's complement of the answer

ADI 00 // To check sign bit

JM SIGNEDBIT // If S = 1 --> SIGNEDBIT

JP EXPONENTS

SIGNEDBIT: CMA

ADI 01

ADI 80

MOV B,A

EXPONENTS: POP H// HL = 2001

MOV C,M

MVI D,00

MVI A,01

CMP E

JZ ADD1

HLT

ADD1: MOV A,C

ADI 01

MOV C,A

MVI A,00

HLT# ORG 2000H

DB FAH,FBH,A0H,20H

OVERFLOW CASE:

The screenshot displays the 8085 Assembly Language Editor interface. The left pane shows the assembly code, and the right pane shows the status of registers and memory.

Assembly Code:

```

Assembler Disassembler
8085 Assembly Language Editor

ADI 00 // Two Check Sign bit
JM YES1 // If Sign Bit is 1 then --> YES1
RAR // If S = 0 then --> Simply rotating
MOV B,A
JMP NEXTNUMBER

YES1: ADI 80 // To remove Sign bit
      CMC
      RAR
      ADI 80 // To add Sign bit
      MOV B,A

NEXTNUMBER: MOV A,C
            ADI 00 // Two Check Sign bit
            JM YES2 // If Sign Bit is 1 then --> YES1
            RAR // If S = 0 then --> Simply rotating
            MOV C,A
            JMP FINALANSWEROFROTATED

YES2: ADI 80 // To remove Sign bit
      CMC
      RAR
      ADI 80 // To add sign bit
      MOV C,A

FINALANSWEROFROTATED: ADD B
                     MOV B,A
                     MOV A,B // To do 2's complement of the answer
                     ADI 00 // To check sign bit
                     JM SIGNEDBIT // If S = 1 --> SIGNEDBIT
                     JP EXPONENTS

SIGNEDBIT: CMA
           ADI 01
           ADI 80
           MOV B,A

EXPONENTS: POP H // HL = 2001
           MOV C,M
           MVI D,00
           MVI A,01
           CMP E
           JZ ADD1
           HLT

ADD1: MOV A,C
      ADI 01
      MOV C,A
      MVI A,00
      HLT

# ORG 2000H
# DB 03H,03H,40H,60H
  
```

Registers:

| Register | Value | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|-------|---|---|---|---|---|---|---|---|
| Accumulator | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Register B | 50 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Register C | 04 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Register D | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Register E | 01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Register H | 20 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Register L | 01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Memory(M) | 03 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Flag Register:

| Register | Value | S | Z | * | AC | * | P | * | CY |
|---------------|-------|---|---|---|----|---|---|---|----|
| Flag Register | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Memory:

| Type | Value |
|--------------------------|-------|
| Stack Pointer(SP) | 4000 |
| Memory Pointer (HL) | 2001 |
| Program Status Word(PSW) | 0000 |
| Program Counter(PC) | 00ED |
| Clock Cycle Counter | 486 |
| Instruction Counter | 74 |

For SIM instruction:

| SOD | SID | * | R7.5 | MSE | M7.5 | M6.5 | M5.5 |
|-----|-----|---|------|-----|------|------|------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

For RIM instruction:

| SID | I7.5 | I6.5 | I5.5 | IE | M7.5 | M6.5 | M5.5 |
|-----|------|------|------|----|------|------|------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

No. Converter Tool:

| Hexadecimal | Decimal | Binary |
|-------------|---------|--------|
| 0 | 0 | 0 |

Created by : Jubin Mitra

Result is stored in Register B, exponent in Reg C and if there is an overflow condition then in Reg E and in this case, overflow was there. Ergo, Register E has the value 01