

Assignment No. 1

1. WAP to count positive and negative numbers from a series of eight signed numbers (2435H, 0C00H etc.).

Program:

LIST:

DW 0xa315

DW 0x2311

DW 0x3676

DW 0x123a

DW 0xe39f

DW 0x3423

DW 0xbc8f

DW 0x8973

start:

MOV SI, OFFSET LIST ; Moving the list to SI

MOV AX, word [SI] ; Moving 1st number to AX

ADD SI, 0x2 ; Fetching 2nd number

MOV CL, 0x8 ; Initializing counter to 8, since we have 8 numbers

MOV BX, 0x0 ; BX and DX will store the total number of
positive and negative numbers respectively in the
given list

MOV DX, 0x0

AGAIN:

SHL AX, 1 ; Here, the number inside AX is converted to binary
and then Shifted Left by 1 bit
; Hence, the MSB is removed and it is stored in

the CF

; If CF is 1, number is negative

; If CF is 0, number is positive

JC SIGNED

INC BX ; If positive number is found, BX increments by 1

JMP NEXT

SIGNED:

INC DX ; If negative number is found, DX increments by 1

NEXT:

ADD SI, 0x2

MOV AX, word [SI]

DEC CL ; Decrementing counter i.e CL after checking each
number

JNZ AGAIN ; Process continues till CL becomes zero

Output:**8086 Compiler**

The screenshot displays the 8086 Compiler interface. On the left is the **Code Editor** with the following assembly code:

```

1 LIST:
2 DW 0xa315
3 DW 0x2311
4 DW 0x3676
5 DW 0x123a
6 DW 0xe39f
7 DW 0x3423
8 DW 0xbc8f
9 DW 0x8973
10
11 start:
12 MOV SI, OFFSET LIST ; Moving the list to SI
13 MOV AX, word [SI] ; Moving 1st number to AX
14 ADD SI, 0x2 ; Fetching 2nd number
15 MOV CL, 0x8 ; Initializing counter to 8, since we have 8 numbers
16 MOV BX, 0x0 ; BX and DX will store the total number of positive and negative numbers respectively
17 MOV DX, 0x0
18
19 AGAIN:
20 SHL AX, 1 ; Here, the number inside AX is converted to binary and then Shifted Left by 1
21 ; Hence, the MSB is removed and it is stored in the CF
22 ; If CF is 1, number is negative
23 ; If CF is 0, number is positive
24 JC SIGNED
25 INC BX ; If positive number is found, BX increments by 1
26 JMP NEXT
27
28 -

```

On the right side, there are three panels:

- Registers:**

| Reg | H | L |
|-----|----|----|
| A | 00 | 00 |
| B | 00 | 04 |
| C | 00 | 00 |
| D | 00 | 04 |
- Segments:**

| Segment | Address |
|---------|---------|
| SS | 0000 |
| DS | 0000 |
| ES | 0000 |
- Pointers:**

| Pointer | Address |
|---------|---------|
| SP | 0000 |
| BP | 0000 |
| SI | 0012 |
| DI | 0000 |

Below these panels is the **Flags:** section:

| OF | DF | IF | TF | SF | ZF | AF | PF | CF |
|----|----|----|----|----|----|----|----|----|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

At the bottom right is the **Memory** section, showing a table of memory addresses and their contents:

| Address | Content |
|-------------------------------------------------|----------------------------------------------|
| 15 a3 11 23 76 36 3a 12 9f e3 23 34 8f bc 73 89 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |

At the bottom left, there are **Input** and **Output** fields.

2.WAP to find out average of a given string (list of numbers) of data bytes neglecting fractions.

Program:

LIST:

DW 0xab18

DW 0x0324

DW 0x03fa

start:

```
MOV SI, OFFSET LIST ; Store the list in SI
MOV CL, 0x3          ; Initialize counter
MOV BX, 0x0          ; BX will store the no. of elements present in the list
MOV AX, 0x0
```

A:

```
ADD AX, word [SI] ; AX stores the addition of numbers
INC BX
ADD SI, 0x2       ; Fetching next number into SI
DEC CL           ; Decrement counter
JNZ A            ; Process continues till CL becomes zero
```

B:

```
DIV BX           ; Dividing AX by BX
                ; It gives the average and stores it in AX
```

Output:**8086 Compiler**

8086 Compiler interface showing assembly code, registers, flags, and memory.

Code Editor

```
1 LIST:
2 DW 0xab18
3 DW 0x0324
4 DW 0x03fa
5
6 start:
7 MOV SI, OFFSET LIST ; Store the list in SI
8 MOV CL, 0x3 ; Initialize counter
9 MOV BX, 0x0 ; BX will store the no. of elements present in the list
10 MOV AX, 0x0
11
12 A:
13 ADD AX, word [SI] ; AX stores the addition of numbers
14 INC BX
15 ADD SI, 0x2 ; Fetching next number into SI
16 DEC CL ; Decrement counter
17 JNZ A ; Process continues till CL becomes zero
18
19 B:
20 DIV BX ; Dividing AX by BX
21 ; It gives the average and stores it in AX
```

Registers

| Reg | H | L |
|-----|----|----|
| A | 3b | 67 |
| B | 00 | 03 |
| C | 00 | 00 |
| D | 00 | 01 |

Segments

| Segment | Address |
|---------|---------|
| SS | 0000 |
| DS | 0000 |
| ES | 0000 |

Pointers

| Pointer | Address |
|---------|---------|
| SP | 0000 |
| BP | 0000 |
| SI | 0006 |
| DI | 0000 |

Flags

| OF | DF | IF | TF | SF | ZF | AF | PF | CF |
|----|----|----|----|----|----|----|----|----|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

Memory

Start Address: 00000

| Address | Value |
|----------------------------------------------|-------|
| 18 ab 24 03 fa 03 00 00 00 00 00 00 00 00 00 | |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |

Input

Output