CLASS:S.E. COMP **SUBJECT: MPL**

EXPT. NO.: 5 **DATE: 28/05/21**
ROLL NO.: 21129

TITLE: To find positive and negative numbers

PROBLEM STATEMENT:
Write X86/64 ALP to count the number of positive and negative numbers from the array.

OBJECTIVE :
To understand how to identify a positive number or negative number

OUTCOME:
Students will be e able to use different index registers and find positive and negative numbers from array

THEORY :

Any number above zero is a positive number. Positive numbers are written with no sign or a + sign in front of them and they are counted from zero to the right. Any number below zero is a negative number. Negative numbers are written with a - sign in front of them and they are counted from zero to the left. Always look at the sign in front of a number to see if it is positive or negative.

In Computer Systems, the negative number is represented by different ways:

1. Sign-Magnitude representation: - Negative numbers are essential and any computer not capable of dealing with them would not be particularly useful. But how can such numbers be represented? There are several methods which can be used to represent negative numbers in Binary. One of them is called the Sign-Magnitude Method. The Sign-Magnitude Method is quite easy to understand. In fact, it is simply an ordinary binary number with one extra digit placed in front to represent the sign. If this extra digit is a '1', it means that the rest of the digits represent a negative number. However if the same set of digits are used but the extra digit is a '0', it means that the number is a positive one. The following examples explain the Sign-Magnitude method better. Let us assume that we have an 8-bit register. This means that we have 7 bits which represent a number and the other bit to represent the sign of the number (the Sign Bit). This is how numbers are represented:

0	0	1	0	0	1	0	1

The red digit means that the number is positive. The rest of the digits represent 37. Thus, the above number in sign-magnitude representation means +37. And this is how -37 is represented:

1	0	1	0	0	1	0	1

Binary	Value
0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

Assign the leftmost (most significant) bit to be the sign bit. If the sign bit is 0, this means the number is positive. If the sign bit is 1, then the number is negative. The remaining m-1 bits are used to represent the magnitude of the binary number in the unsigned binary notation.

2. **1's Complement representation: -** The 1‟s complement of a binary number is the number that results when we change all 1s to zeros and the zeros to ones.

3. **2's Complement representation: -** Another common method of representing negative numbers in binary is the Two's Complement. The 2‟s complement is the binary number that results when we add 1 to the 1‟s complement. It is given as 2‟s complement = 1‟s complement + 1

2's complement = 1's complement + 1

e.g. 2's complement of $(11000100)_2$

1's complement of number = 00111011

Add 1 = 1

2's complement of number = 00111100

A complete binary table for four bits is shown below:

Binary	Two's Comp Value	Binary	Two' Comp Value
0000	0	1111	-1
0001	+1	1110	-2
0010	+2	1101	-3
0011	+3	1100	-4
0100	+4	1011	-5
0101	+5	1010	-6
0110	+6	1001	-7
0111	+7	1000	-8

Implementation and Algorithm:

1. Start.
2. Initialize an array of 10 numbers or accept 10 numbers from the user and store them in one array.
3. Initialize pos_counter=0, neg_counter=0, index_reg=array address, counter=10
4. Read the number from index_reg into a register.
5. Perform addition with 00H and check sign bit
6. If sign bit==1 then increment neg_counter=neg_counter+1 else increment
   pos_counter=pos_counter+1 end if

7. Increment index_reg= index_reg+1
8. Decrement counter=counter-1
9. If counter!=0 then goto step number 4 else P:F-LTL-UG/03/R1 continue
10. Print message “Positive numbers are:” and print pos_counter.
11. Print message “Negative numbers are:” and print neg_counter.
12. Exit

Test Cases Executed:

1. Input array: 12, 23, -45, -67, 90, -66, 45, -67, -5, -45

   Output: positive: 4

   negative: 6
2. Input array: 2,3,5,7,-5,-7,-5,-,6

   Output: positive: 4

   Negative: 4

CONCLUSION:

We are able to use different index registers and find positive and negative numbers from the array.