Batch: E1

Microprocessor Lab Assignment

Problem Statement:

Write x86/64 ALP to count number of positive and negative numbers from array.

Objectives:

To implement a ALP for counting positive and negative numbers with help of instructions in an array.

Outcomes:

We will be able to segregate positive and negative numbers and keep count of them to display them.

Theory:

Sign Magnitude Representation:

It is represented as an ordinary binary number with one extra digit placed in front of MSB to represent sign. If this extra digit is '1', it means that rest of digits represent a negative number. If extra digit is '0', rest of digit represents positive number.

Eg. 00100101 ⇒ positive number

11010110 ⇒ negative number

Manas 21121

2's complement representation:
2's complement is binary number that results when we add 1 to the 1's complement.
Eg. 2's complement of $(110010)_2$
1's complement = 001101
2's complement = 1's complement + 1
= 001101 + 1
= $(001110)_2$

Directives used: section.data, section.text, section.bss
System calls used: read, write and exit

Instructions used:
i) rcl (Rotate carry left): Rotates all of the bits in specified word or byte some number of positions left along with carry flag.
Flags affected: CF
Eg. mov cl, 04h
rcl al, cl ; rotate al 4 bits left

ii) jnc (Jump if not carry): Checks whether carry flag is set or not. If yes [CF=0] then jumps to label mentioned.
Eg. jnc l

iii) and: Performs logical AND operation of two operands. Flags Affected: OF, CF, SF and PF
Eg. and al, 1BH

Manas 21121

iv) jbe (jump below or equal): Jumps to the destination label mentioned if concerned value is below or equal to another value.
Flags affected: CF, ZF
Eg. l1: cmp bl, 05h
jbe l1

v) rol (rotate left): Rotates all the bits in the register towards left
Flags affected: OF, CF

vi) cmp (compare): Subtracts source operand from destination operand but doesn't alter either of them.
Flags affected: OF, SF, ZF, AF, PF, CF
Eg. cmp bl, 05h

Algorithm:

1. Start
2. Point Register towards numbers in memory location.
3. Move value of array pointer to another register.
4. Rotate register through carry left by 1.
5. If carry flag is set, increase count of negative numbers by 1 or else increase count of positive numbers by 1.
6. Increment array pointer by 1 byte.
7. Decrement loop count
8. If counter isn't equal to zero, jump to label.

Manas 21121

- Conversion of hex to ASCII for printing.

9. Rotate the contents of register storing count of positive numbers by 4 bit to left.
10. Move this number to another register to perform logical AND operation with 0Fh.
11. Compare result with 09h, If it is less than equal to 09h, add 30h otherwise add 37h.
12. Move resultant to array storing ASCII for displaying proper result.
13. Increment ASCII array pointer.
14. Decrement counter
15. If counter is not equal zero jump to step 9.
16. Repeat steps 9 to 15 for count of negative numbers.
17. Print ASCII values of both positive and negative numbers on screen.
18. Exit

	Input	Expected Output	Actual Output	Status
1.	03h, -55h, 30h, -40h, 75h	Positive = 3 Negative = 2	Positive = 3 Negative = 2	Pass
2.	04h, -30h, -40h, -75h, 55h	Positive = 2 Negative = 3	Positive = 2 Negative = 3	Pass

Conclusion: We were successfully able to count number of positive and negative numbers from an array and print output by undergoing hex to ASCII conversion with use of loops, registers and decision statements.