ASSIGNMENT NO. 1

AIM :- Write X86/64 ALP to count number of positive and negative numbers from the array

APPARATUS :

- Core 2 duo/i3/i5/i7 - 64bit processor
- OS – ubuntu 32bit/64bit OS
- Assembler used –nasm
- Editor Used – gedit

THEORY :

Rotate Instructions :-

In 8086 you have two choices capable to perform single rotation :

1. The number of bits, to be rotated, normally referred as COUNT, may be specified as the constant 1
2. COUNT may be specified in register CL. Register CL is by default register used, for this operation.

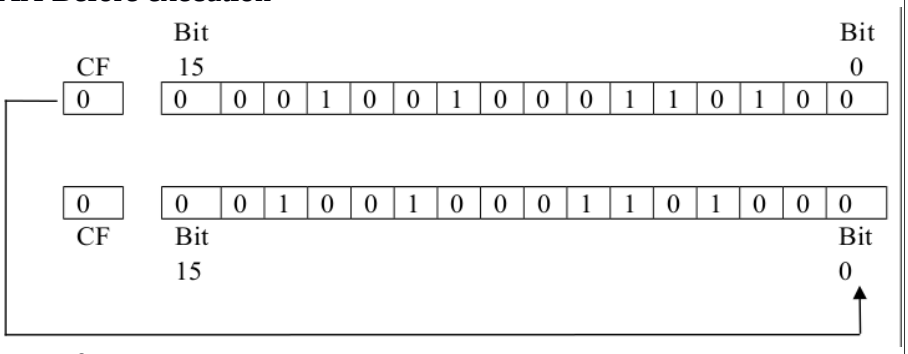
Table 1 summarizes all rotate instruction.

Table 1: ROTATE Instructions

Instruction	Description
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

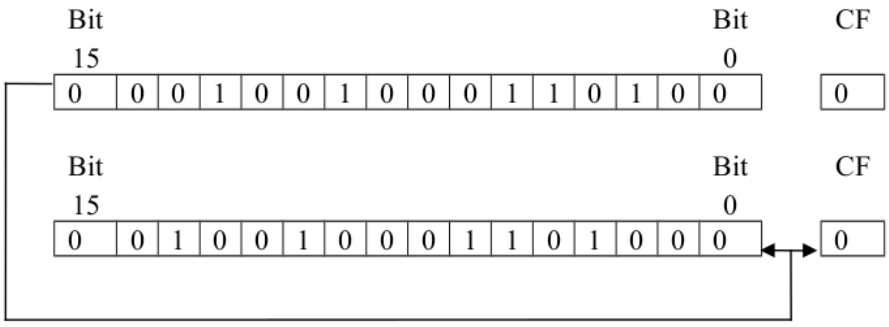
RCL – Rotate through Carry Left

Description	Rotate through Carry Left

Mnemonic	RCL destination, count
Algorithm	Shift all bits left, the bit that goes is set to CF and previous value of CF is inserted to the rightmost position.
Operation	• This instruction rotates all the bits in specified destination by some number of bit positions to the left. The destination can be register or memory location. • Negative shifts are illegal. • CL is default register used for rotate instruction when count is greater than 1.
Flags	Only CF and OF are affected
Example	RCL AX, 01H AX Before execution  AX After execution

ROL – Rotate Left

Description	Rotate Left
Mnemonic	ROL destination, source
Algorithm	Shift all bits left, the bit that goes off is set to CF and the same bit is inserted to the rightmost position.
Operation	• This instruction rotates all the bits in specified word or byte to the left by some bit positions. • The data bit rotated out of MSB is circled back into the LSB. The data bit otated out of MSB is also copied to CF.

	• CL is default register used for rotate instruction when count is greater han 1.
Flags	Only CF and OF are affected
Example	RCL AX, 01H AX Before execution  AX After execution

ALGORITHM:-

The algorithm for implementation of 64 bit program is as below :-

1. Start
2. Declare & initialize the variables in .data section.
3. Declare uninitialized variables in .bss section.
4. Declare Macros for print and exit operation.
5. Initialize pointer with source address of array.
6. Initialize count for number of elements.
7. Set RBX as Counter register for storing positive numbers count.
8. Set RCX as Counter register for storing negative numbers count.
9. Get the number in RAX register.
10. Rotate the contents of RAX register left 1 bit to check sign bit.
11. Check if MSB is 1. If yes, goto step 12,else goto step 13.
12. Increment count for counting negative numbers.

13. Increment count for counting positive numbers.
14. Increment pointer.
15. Decrement count
16. Check for count = 0. If yes, goto step 17 else goto step 9.
17. Store the positive numbers count and negative numbers count in buffer.
18. Display first message. Call the procedure to display the positive count.
19. Display second message. Call the procedure to display the negative count.
20. Add newline.
21. Terminate the process.
22. Declare the Procedure.
23. Stop

CONCLUSION

_____
_____
_____
_____