

Pune Institute of Computer Technology, Pune
Department of Computer Engineering

Sub :- MPL

Date:- 05/06/2021

Name: - Shrikrushna Santosh Zirape

Roll No: - 21286

Assignment No 4

Title :-

To find positive and negative numbers.

Problem Statement :-

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

Sw/Hw requirements :-

Intel i5 8th generation 64 bit processor

OS - Ubuntu 16.0 LTS

Editor : - VS- Code , Gedit

Assembler: Nasm

Debugger : gdb

Objective :-

To understand how to identify a positive number or negative number

Outcome :-

Students will be 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.

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 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

Advantages of 2's complement representation: -

1. There are distinct +0 and -0 representations in both the sign-magnitude and 1's complement systems, but the 2's complement system has only a +0 representation.
2. For 4 bit numbers, the value -8 is representable only in the 2's complement system and not in other systems.
3. It is more efficient for logic circuit implementation, and one often used in computers for addition and subtraction operations.

System calls :

Write System Call :

Mov rax, 1

Mov rdi , 1

Mov rsi, msg

Mov rdx, 0BH

Syscall

Read System Call :

Mov rax, 1

Mov rdi , 1

Mov rsi, msg

Mov rdx, 0BH

Syscall

Exit System call

Mov rax, 60

Mov rsi, 00

syscall

Instructions Used :-

1. Add:

Description: This instruction adds a number from source to number from destination and puts the result to specified destination.

destination=destination+source

Flags: CF, ZF, OF, PF

e.g. add eax, ebx

2. jnz:

Description: This instruction is used to jump to next instruction in the program

when the zero flag is not equal to 0.

Flags: Only the ZF is affected.

Example: JNZ L1

3. DEC:

Description: This instruction subtracts 1 from the destination word, double word or byte.

Flags: SF, ZF, OF, PF and AF are affected.

Example: DEC AL

4. INC:

Description: This instruction adds 1 to the destination operand.

Flags: SF, PF, OF, ZF, AF are affected.

Example: INC CX

5. BT:

Description: This instruction tests the status the specified bit

in the instruction. The status of that bit is copied to the carry flag.

Flags: CF is set to the value of selected bit and OF, ZF, SF, AF and PF are undefined.

Example: BT EAX, 05

Algorithm :-

1. Start.
2. Initialize an array of 10 numbers
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 bit test and check MSB
6. If msb==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
continue
10. Print message "Positive numbers are:" and print pos_counter.
11. Print message "Negative numbers are:" and print neg_counter.
12. Exit.

Conclusion :-

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

Program : -

```
%macro print 2
```

```
mov rax, 01
```

```
mov rdi, 01
```

```
mov rsi, %1
```

```
mov rdx, %2
```

```
syscall
```

```
%endmacro
```

```
section .data
```

```
arr dq 111111111111, 222222222222222, 33333333333333, 4444444444444,  
5555555555555, 6666666666666, 777777777777
```

```
msg1 db "positive no's : "
```

```
msg2 db "negative no's : "
```

```
nl db "",10
```

```
section .bss
```

```
tcount resb 1
```

```
pcount resb 1
```

```
ncount resb 1
```

```
temp resb 1
```

```
section .text
```

```
global _start
```

_start:

mov byte[tcount], 7

mov byte[pcount], 0

mov byte[ncount], 0

mov rsi, arr

mov rcx, tcount

u1:

bt qword[rsi], 63

jnc pnext

inc byte[ncount]

jmp skip

pnext:

inc byte[pcount]

skip:

inc rsi

dec byte[tcount]

jnz u1

print msg1, 16


```
mov al, byte[pcount]
```

```
call _byteprint
```

```
print nl, 1
```

```
print msg2, 16
```

```
mov al, byte[ncount]
```

```
call _byteprint
```

```
print nl, 1
```

```
mov rax, 60
```

```
mov rdi, 00
```

```
syscall
```

```
_byteprint:
```

```
mov esi, temp
```

```
; mov al, 9dh
```

```
mov cl, 02
```

```
l:
```

```
rol al, 04
```

```
mov bl, al
```

```
and bl, 0Fh
```

```
cmp bl, 09h
```

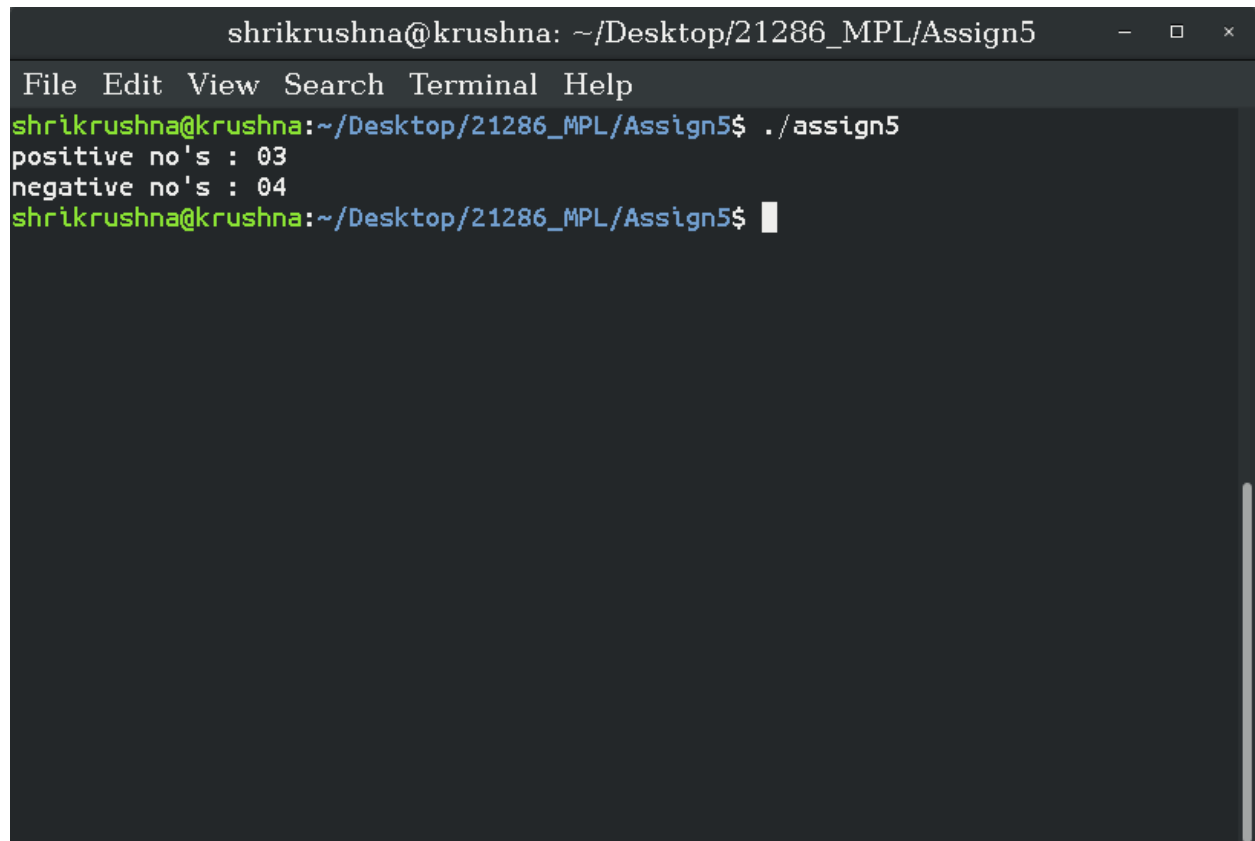
```
jbe m
```

```
add bl, 07h
```

```
m:
```

```
add bl, 30h
mov [esi], bl
inc esi
dec cl
jnz l
print temp, 02
ret
```

OUTPUT:-

A screenshot of a terminal window with a dark background. The title bar at the top reads "shrikrushna@krushna: ~/Desktop/21286_MPL/Assign5" and includes standard window control buttons. The terminal menu bar shows "File Edit View Search Terminal Help". The command prompt is "shrikrushna@krushna:~/Desktop/21286_MPL/Assign5\$". The user has entered the command "./assign5", and the terminal has outputted two lines: "positive no's : 03" and "negative no's : 04". The prompt is now ready for the next command.

```
shrikrushna@krushna: ~/Desktop/21286_MPL/Assign5
File Edit View Search Terminal Help
shrikrushna@krushna:~/Desktop/21286_MPL/Assign5$ ./assign5
positive no's : 03
negative no's : 04
shrikrushna@krushna:~/Desktop/21286_MPL/Assign5$
```