Assembly Programs:

1. Write an alp to enter elements in array and display its contents.

section .data

msg db 'Enter number of elements: '

msglen equ $-msg

msg1 db 'Enter elements:'

msg1len equ $-msg1

msg2 db 'Array elements: '

msg2len equ $-msg2

msg3 db ' Enter Array elements: ', 10

msg3len equ $-msg3

newline db '',10

space db ' '

%macro write 2

mov eax,4

mov ebx,1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

mov eax,3

mov ebx,2

mov ecx,gb

mov edx,1

int 80h

%endmacro

input\_array:

mov byte[i], 0

mov esi, arr

ip\_loop:

read element,1

mov ebx, [element]

sub ebx, '0'

mov [esi], ebx

inc esi

inc byte[i]

mov al, [i]

mov bl, [n]

cmp al, bl

jl ip\_loop

ret

display\_array:

mov byte[i], 0

mov esi, arr

op\_loop:

mov eax, [esi]

add eax, '0'

mov [element], eax

write element, 1

write space, 1

inc esi

inc byte[i]

mov al, [i]

mov bl, [n]

cmp al, bl

jl op\_loop

write newline,1

ret

section .bss

n RESB 1

arr resb 10

i resb 1

gb resb 1

element resb 1

section .text

global \_start

\_start:

write msg, msglen

read n, 1

mov eax, [n]

sub eax, '0'

mov [n], eax

write msg3, msg3len

call input\_array

write msg2 ,msg2len

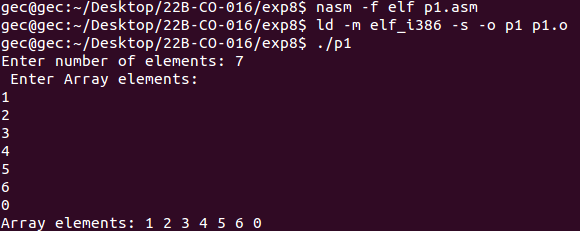
call display\_array

mov eax, 1

mov ebx, 0

int 80h

**Output:**



2. Write an alp to count no. of positive, negative and zeros

section .data

msg1 db 'Number of Positive numbers: '

msg1len equ $-msg1

msg2 db 'Number of Negative numbers: '

msg2len equ $-msg2

new db '',10

nl equ $-new

arr db 1, -2, -3, -11, 5, -6, 4

%macro write 2

mov eax,4

mov ebx,1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

fun:

mov eax, 0

mov [p\_count], eax

mov [n\_count], eax

mov ecx, 7

mov esi, arr

label:

mov bl, [esi]

cmp bl,0

jg positive

inc byte[n\_count]

jmp end

positive:

inc byte[p\_count]

end:

inc esi

loop label

mov eax, [p\_count]

add eax, '0'

mov [p\_count], eax

mov eax, [n\_count]

add eax, '0'

mov [n\_count], eax

ret

section .bss

p\_count resb 1

n\_count resb 1

section .text

global \_start

\_start:

call fun

write msg1,msg1len

write p\_count, 1

write new,nl

write msg2,msg2len

write n\_count, 1

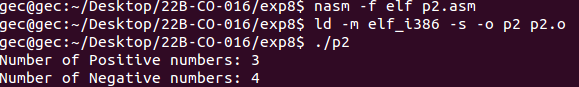
write new,nl

mov eax ,1

mov ebx ,0

int 80h

**Output:**

****

3. Write an alp to count the number of odd and even numbers

section .data

msg1 db 'Enter the elements of the array:', 10

msg1len equ $-msg1

msg2 db 'Even number of elements:'

msg2len equ $-msg2

msg3 db 'Enter the number of elements in the array: '

msg3len equ $-msg3

msg4 db 'Odd number of elements:'

msg4len equ $-msg4

newline db ' ',10

newlinelen equ $-newline

tab db ''

tablen equ $-tab

array dw 0,0,0,0,0,0,0,0,0

arraylen equ 9

%macro write 2

mov eax,4

mov ebx,1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

mov eax,3

mov ebx,2

mov ecx,gb

mov edx,1

int 80h

%endmacro

section .bss

n resb 9

element resb 9

odd resb 9

even resb 9

rem resb 9

i resb 9

gb resb 1

section .text

global \_start

\_start:

write msg3, msg3len

read n, 1

write msg1, msg1len

mov eax, '0'

mov [even], eax

mov eax, '0'

mov [odd], eax

mov eax, [n]

mov [i], eax

mov esi, array

mov eax,n

input\_element:

read element, 1

mov bx,[element]

mov [esi], bx

dec byte[n]

inc esi

cmp byte[n], '0'

jne input\_element

mov esi, array

mov eax,i

check:

mov eax, [esi]

mov [element], eax

write tab, tablen

mov al,[element]

sub al,'0'

mov bl,'2'

sub bl,'0'

div bl

add ah,'0'

add al,'0'

mov[rem],ah

mov eax,[rem]

mov ebx,'0'

cmp eax,ebx

je EVEN

jne ODD

EVEN:

inc byte[even]

jmp CMP

ODD:

inc byte[odd]

jmp CMP

CMP:

dec byte[i]

inc esi

cmp byte[i],'0'

jne check

write msg2, msg2len

write tab, tablen

write even, 1

write newline, newlinelen

write msg4, msg4len

write tab, tablen

write odd, 1

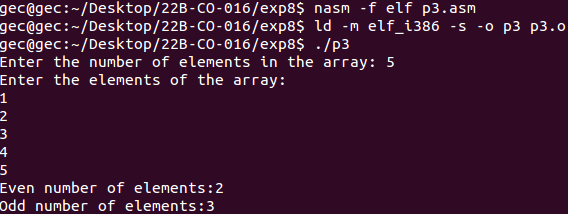
write newline, newlinelen

mov eax,1

mov ebx,0

int 80h

**Output:**



4. Write an alp to count the number of elements greater lesser and equal

section .data

msg1 db 'Enter the elements of the array:', 10

msg1len equ $-msg1

msg2 db 'Enter the number of elements in the array: '

msg2len equ $-msg2

msg3 db 'The number of elements above 5 are'

msg3len equ $-msg3

msg4 db 'The number of elements below 5 are'

msg4len equ $-msg4

newline db ' ',10

newlinelen equ $-newline

tab db ' '

tablen equ $-tab

;array declaration and initialization

array dw 0,0,0,0,0,0,0,0,0

arraylen equ 9

%macro write 2

mov eax,4

mov ebx,1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

mov eax,3

mov ebx,2

mov ecx,gb

mov edx,1

int 80h

%endmacro

section .bss

n resb 9

element resb 9

i resb 9

above resb 9

below resb 9

gb resb 1

section .text

global \_start

\_start:

write msg2, msg2len

read n, 1

write msg1, msg1len

mov eax, [n]

mov [i], eax

mov esi, array

mov eax, n

input\_element:

read element, 1

mov ebx, [element]

mov [esi], ebx

dec byte[n]

inc esi

cmp byte[n], '0'

jne input\_element

mov esi, array

mov eax,i

mov byte[above],'0'

mov byte[below],'0'

check:

mov eax, [esi]

mov [element], eax

mov al, [element]

mov bl,'5'

cmp al,bl

jl LESS

jg GREATER

LESS:

inc byte[below]

jmp END

GREATER:

inc byte[above]

jmp END

END:

dec byte[i]

inc esi

cmp byte[i],'0'

jne check

write msg3, msg3len

write above, 1

write tab, tablen

write newline, newlinelen

write msg4, msg4len

write below, 1

write tab, tablen

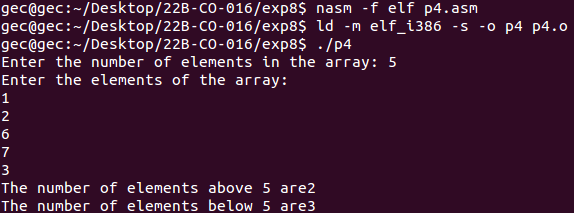
write newline, newlinelen

mov eax, 1

mov ebx,0

int 80h

**Output:**

****

5. Write an alp to find the sum of the elements of an array

section .data

msg1 db 'Enter the elements of the array:', 10

msg1len equ $-msg1

msg2 db 'Enter the number of elements in the array: '

msg2len equ $-msg2

msg3 db 'The sum is:'

msg3len equ $-msg3

newline db ' ',10

newlinelen equ $-newline

tab db ' '

tablen equ $-tab

;array declaration and initialization

array dw 0,0,0,0,0,0,0,0,0

arraylen equ 9

%macro write 2

mov eax,4

mov ebx,1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

mov eax,3

mov ebx,2

mov ecx,gb

mov edx,1

int 80h

%endmacro

section .bss

n resb 9

element resb 9

i resb 9

sum resb 1

gb resb 1

section .text

global \_start

\_start:

write msg2, msg2len

read n, 1

write msg1, msg1len

mov eax, [n]

mov [i], eax

mov esi, array

mov eax, n

input\_element:

read element, 1

mov ebx, [element]

mov [esi], ebx

dec byte[n]

inc esi

cmp byte[n], '0'

jne input\_element

mov eax,'0'

mov [sum],eax

mov esi, array

mov eax, i

SUM:

mov eax,[esi]

mov [element],eax

mov al,[element]

sub al,'0'

mov bl,[sum]

sub bl,'0'

add al,bl

add al,'0'

mov [sum],al

dec byte[i]

inc esi

cmp byte[i],'0'

jne SUM

write msg3, msg3len

write sum,1

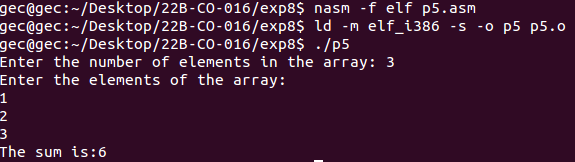
write newline, newlinelen

mov eax,1

mov ebx,0

int 80h

**Output:**

****