**Code:**

%macro write 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

%macro read 2

mov eax, 3

mov ebx, 2

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

section .data

asksize db 'Enter size: '

asksizelen equ $-asksize

ask db 'Enter array: '

asklen equ $-ask

show db 'Array contents: '

showlen equ $-show

array times 20 dw 0

len equ 20

nl db '', 10

nllen equ $-nl

spa db ' '

spalen equ $-spa

section .bss

num resb 2

i resb 2

element resb 2

section .text

global \_start:

\_start:

write asksize, asksizelen

read num, 2

write ask, asklen

write nl, nllen

mov byte[i], 0

mov esi, array

input:

read element, 2

mov ebx, [element]

mov [esi], ebx

inc esi

inc byte[i]

mov al, [i]

mov bl, [num]

sub bl, '0'

cmp al, bl

JE exit\_input

JMP input

exit\_input:

write show, showlen

write nl, nllen

mov byte[i], 0

mov esi, array

output:

mov ebx, [esi]

mov [element], ebx

write element, 1

write spa,spalen

inc esi

inc byte[i]

mov al, [i]

mov bl, [num]

sub bl, '0'

cmp al, bl

JL output

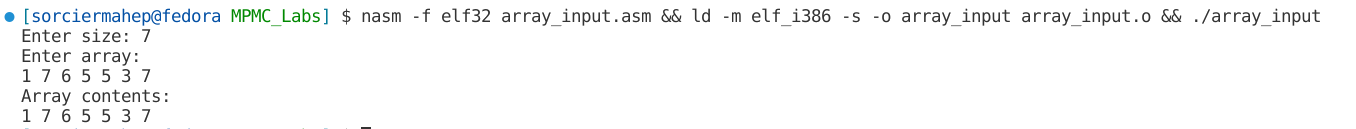
write nl,nllen

mov eax, 1

xor ebx,ebx

int 80h

**Output:**

****

**Code:**

%macro write 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

%macro read 2

mov eax, 3

mov ebx, 2

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

section .data

arraymsg db "Array contents are 1, 2, -7, 11, -10, 4, 8, 2, 5",10

arraymsgstr equ $-arraymsg

str1 db "Array is: ",10

str1len equ $-str1

pos\_msg db "Positive count:"

pos\_len equ $-pos\_msg

neg\_msg db "Negative count:"

neg\_len equ $-neg\_msg

nl db '', 10

nllen equ $-nl

array dw 1, 2, -7, 11, -10, 4, 8, 2, 5

arrcnt equ 9

poscnt dw 0

negcnt dw 0

section .bss

dis\_buffer resb 2

section .text

global \_start

\_start:

write arraymsg,arraymsgstr

mov esi, array

mov ecx, arrcnt

UP:

BT word[esi], 15

JC NEG

inc byte[poscnt]

JMP POS

NEG: inc byte[negcnt]

POS:

inc esi

inc esi

loop UP

write pos\_msg, pos\_len

mov bl, [poscnt]

CALL HEX\_ASCII

write nl, nllen

write neg\_msg, neg\_len

mov bl, [negcnt]

CALL HEX\_ASCII

write nl, nllen

mov eax, 1

xor ebx,ebx

int 0x80

HEX\_ASCII:

mov ecx, 2

mov edi, dis\_buffer

DUP:

rol bl, 04

mov al, bl

and al, 0fh

cmp al, 09h

jbe NEXT

add al, 07h

NEXT:

add al, 30h

mov [edi], al

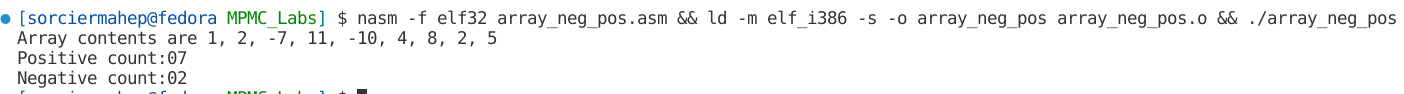
inc edi

loop DUP

write dis\_buffer, 2

ret

**Output:**

****

**Code:**

%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

%endmacro

section .data

asknum db 'Enter the number of elements: '

asknumlen equ $-asknum

ask db 'Enter the array: '

asklen equ $-ask

showeven db 'Even count: '

showevenlen equ $-showeven

showodd db 'Odd count: '

showoddlen equ $-showodd

nl db '', 10

nllen equ $-nl

array times 20 dw 0

len equ 20

section .bss

num resb 10

i resb 10

el resb 10

rem resb 5

neven resb 5

nodd resb 5

section .text

global \_start

\_start:

write asknum, asknumlen

read num, 10

write ask, asklen

mov byte[neven], 0

mov byte[nodd], 0

mov byte[i], 0

mov esi, array

input:

read el, 2

mov ebx, [el]

mov [esi], ebx

inc esi

inc byte[i]

mov al, [i]

mov bl, [num]

sub bl, '0'

cmp al, bl

jl input

mov byte[i], 0

mov esi, array

check:

mov al, [esi]

mov bl, '2'

sub bl, '0'

div bl

cmp ah, 0

JE even

JMP odd

even:

inc byte[neven]

jmp loop

odd:

inc byte[nodd]

jmp loop

loop:

inc esi

inc byte[i]

mov al, [i]

mov bl, [num]

sub bl, '0'

cmp al, bl

JL check

JE output

output:

add [neven], byte '0'

add [nodd], byte '0'

write showeven, showevenlen

write neven, 5

write nl, nllen

write showodd, showoddlen

write nodd, 5

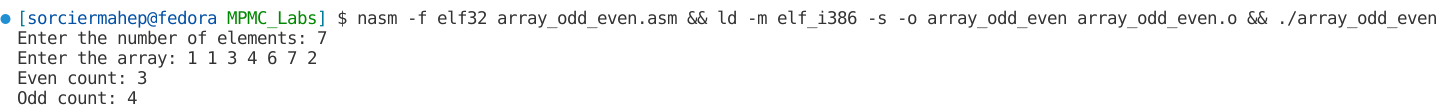
write nl, nllen

mov eax, 1

xor ebx,ebx

int 80h

**Output:**

****

**Code:**

section .data

arraymsg db 'Array contents are: 2,10,73,92,30,56',10

arraymsglen equ $-arraymsg

msg1 db 'Numbers above 50: '

msg1len equ $-msg1

msg2 db 'Numbers below 50: '

msg2len equ $-msg2

newline db '',10

n1 equ $-newline

%macro write 2

mov eax,4

mov ebx,1

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 0x80

%endmacro

global arr

arr: db 2,10,73,92,30,56

section .bss

above resb 1

below resb 1

temp resb 5

section .text

global \_start

\_start:

write arraymsg,arraymsglen

call count

write msg1,msg1len

write above, 1

write newline,n1

write msg2,msg2len

write below, 1

write newline,n1

mov eax ,1

xor ebx,ebx

int 0x80

count:

mov eax, 0

mov [above],eax

mov [below], eax

mov ecx, 6

mov esi, 0

L1:

movzx edi, byte[arr+esi]

mov [temp],edi

mov al,[temp]

mov bl,50

cmp al, bl

jg L2

inc byte[below]

jmp end

L2:

inc byte[above]

end:

inc esi

loop L1

mov eax, [below]

add eax, '0'

mov [below], eax

mov eax, [above]

add eax, '0'

mov [above], eax

ret

**Output:**

