

## 1] Write an ALP to input and display array elements

### INPUT –

```
%macro print 2
    mov eax, 4
    mov ebx, 1
    mov ecx, %1
    mov edx, %2
    int 80h
%endmacro

%macro read 2
    mov eax, 3
    mov ebx, 0
    mov ecx, %1
    mov edx, %2
    int 80h
%endmacro

section .data
    msg db 'Enter the elements of the array : ', 0xA
    plen equ $ - msg
    msg1 db 'The array elements are: '
    mlen equ $ - msg1
    space db ' '
    slen equ $ - space
    newline db 0xA
    nlen equ $ - newline

section .bss
    array resb 5
    num resb 2

section .text
    global _start

_start:
    mov ecx, 5
    mov esi, 0

    push ecx
    print msg, plen
    pop ecx

input_loop:
    push ecx
    read num, 2
    mov al, [num]
    sub al, '0'
    mov [array + esi], al
    inc esi
    pop ecx
    loop input_loop

    print msg1, mlen

    mov ecx, 5
    mov esi, 0

display_loop:
    push ecx
    mov al, [array + esi]
```

add al, '0'	loop display_loop
mov [num], al	
print num, 1	print newline, nlen
cmp ecx, 1	
je skip_space	mov eax, 1
print space, slen	mov ebx, 0
skip_space:	int 80h
inc esi	
pop ecx	

## OUTPUT –

```

root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp8# ./1
Enter the elements of the array :
6
5
7
2
1
The array elements are: 6 5 7 2 1

```

## 2]Write an ALP to count number of positive and negative numbers in an array

### INPUT-

```
%macro print 2                                pos_count resb 1
    mov eax, 4                                neg_count resb 1
    mov ebx, 1
    mov ecx, %1                                section .text
    mov edx, %2                                global _start
    int 80h
%endmacro                                     _start:
                                                mov ecx, 5
                                                mov esi, 0
%macro read 2                                    mov byte[pos_count], 0
    mov eax, 3                                mov byte[neg_count], 0
    mov ebx, 0
    mov ecx, %1
    mov edx, %2                                push ecx
    int 80h                                    print msg, plen
%endmacro                                     print newline, nlen
                                                pop ecx

section .data
    msg db 'Enter the elements (with sign): '
    plen equ $ - msg
    pos_msg db 'Positive numbers: '
    pos_len equ $ - pos_msg
    neg_msg db 'Negative numbers: '
    neg_len equ $ - neg_msg
    newline db 0xA
    nlen equ $ - newline

    input_loop:
        push ecx
        read num, 3
        mov al, [num]
        cmp al, '-'
        je negative
        sub al, '0'
        inc byte[pos_count]
        jmp store

section .bss
    negative:
        mov al, [num + 1]
        sub al, '0'
```

neg al	print newline, nlen
inc byte[neg_count]	
store:	print neg_msg, neg_len
mov [array + esi], al	mov al, [neg_count]
inc esi	add al, '0'
pop ecx	mov [num], al
loop input_loop	print num, 1
	print newline, nlen
print pos_msg, pos_len	
mov al, [pos_count]	mov eax, 1
add al, '0'	mov ebx, 0
mov [num], al	int 80h
print num, 1	

## OUTPUT –

```

root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp8# ./2
Enter the elements (with sign):
-6
-2
5
1
-8
Positive numbers: 2
Negative numbers: 3

```

### 3] Write an ALP to count number of even and odd numbers in the array

#### INPUT –

```
%macro print 2
    mov eax, 4
    mov ebx, 1
    mov ecx, %1
    mov edx, %2
    int 80h
%endmacro

%macro read 2
    mov eax, 3
    mov ebx, 0
    mov ecx, %1
    mov edx, %2
    int 80h
%endmacro

section .data
    msg db 'Enter the elements of the array: ', 0xA
    plen equ $ - msg
    msg1 db 'Number of even numbers: '
    mlen1 equ $ - msg1
    msg2 db 'Number of odd numbers: '
    mlen2 equ $ - msg2
    newline db 0xA
    nlen equ $ - newline

section .bss
    array resb 5
    num resb 2

even_count resb 1
odd_count resb 1

section .text
    global _start

_start:
    mov ecx, 5
    mov esi, 0
    mov byte[even_count], 0
    mov byte[odd_count], 0

    push ecx
    print msg, plen
    pop ecx

input_loop:
    push ecx
    read num, 2
    mov al, [num]
    sub al, '0'
    mov [array + esi], al
    inc esi
    pop ecx
    loop input_loop

    mov ecx, 5
    mov esi, 0
count_loop:
```

mov al, [array + esi]	add al, '0'
mov ah, 0	mov [num], al
mov bl, 2	print num, 1
div bl	print newline, nlen
cmp ah, 0	
jz even_number	print msg2, mlen2
inc byte[odd_count]	mov al, [odd_count]
jmp next_number	add al, '0'
even_number:	mov [num], al
inc byte[even_count]	print num, 1
next_number:	print newline, nlen
inc esi	
loop count_loop	mov eax, 1
	mov ebx, 0
print msg1, mlen1	int 80h
mov al, [even_count]	

## OUTPUT –

```

root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp8# ./3
Enter the elements of the array:
6
5
4
3
8
Number of even numbers: 3
Number of odd numbers: 2
root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp8#

```

#### 4] Write an ALP to calculate elements less than 5 in the array

##### INPUT –

```
%macro print 2                                array resb 5
    mov eax, 4                                num resb 2
    mov ebx, 1                                countLess resb 1
    mov ecx, %1                                countGreater resb 1
    mov edx, %2                                countEqual resb 1
    int 80h
%endmacro

%macro read 2
    mov eax, 3
    mov ebx, 0
    mov ecx, %1
    mov edx, %2
    int 80h
%endmacro

section .text
    global _start

_start:
    mov ecx, 5
    mov esi, 0
    push ecx
    print msg, plen
    pop ecx

section .data
    msg db 'Enter the elements of the array : ', 0xA
    plen equ $ - msg
    msg1 db 'Count of elements less than 5: '
    mlen equ $ - msg1
    msg2 db 'Count of elements greater than 5: '
    m2len equ $ - msg2
    msg3 db 'Count of elements equal to 5: '
    m3len equ $ - msg3
    newline db 0xA
    nlen equ $ - newline

section .bss
    input_loop:
        push ecx
        read num, 2
        mov al, [num]
        sub al, '0'
        mov [array + esi], al
        inc esi
        pop ecx
        loop input_loop

        mov ecx, 5
        mov esi, 0
        mov byte[countLess], 0
```

mov byte[countGreater], 0	mov [num], al
mov byte[countEqual], 0	print num, 1
	print newline, nlen
count_loop:	
mov al, [array + esi]	print msg2, m2len
cmp al, 5	mov al, [countGreater]
je equal_to_5	add al, '0'
jg greater_than_5	mov [num], al
inc byte[countLess]	print num, 1
jmp continue_count	print newline, nlen
greater_than_5:	
inc byte[countGreater]	print msg3, m3len
jmp continue_count	mov al, [countEqual]
equal_to_5:	add al, '0'
inc byte[countEqual]	mov [num], al
continue_count:	print num, 1
inc esi	print newline, nlen
loop count_loop	
	mov eax, 1
print msg1, mlen	mov ebx, 0
mov al, [countLess]	int 80h
add al, '0'	

## OUTPUT –

```

root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp8# ./4
Enter the elements of the array :
6
7
5
2
1
Count of elements less than 5: 2
Count of elements greater than 5: 2
Count of elements equal to 5: 1

```



## 5] Write an ALP to find the sum of elements of the array

### INPUT –

```
%macro print 2                                array resd 5
    mov eax, 4                                num resb 6
    mov ebx, 1                                sum resd 1
    mov ecx, %1                                dispbuf resb 6
    mov edx, %2
    int 80h
%endmacro                                     section .text
                                           global _start

%macro read 2                                _start:
    mov eax, 3                                mov ecx, 5
    mov ebx, 0                                mov esi, 0
    mov ecx, %1
    mov edx, %2                                push ecx
    int 80h                                print msg, plen
%endmacro                                pop ecx

section .data                                input_loop:
    msg db 'Enter the elements of the array: ', 0xA
    plen equ $ - msg                        push ecx
    msg1 db 'The array elements are: '        read num, 6
    mlen equ $ - msg1                        mov ebx, 0
    msg2 db 'Sum of elements: '              mov ecx, 0
    slen equ $ - msg2                        convert:
    space db ' '                            mov al, [num + ecx]
    splen equ $ - space                      cmp al, 0xA
    newline db 0xA                           je done_convert
    nlen equ $ - newline                     sub al, '0'
                                           imul ebx, 10
                                           add bl, al

section .bss
```

inc ecx	mov [ecx], dl
jmp convert	test eax, eax
	jnz convert_to_ascii
done_convert:	
mov [array + esi*4], ebx	print ecx, 6
add esi, 1	print space, splen
pop ecx	add esi, 1
loop input_loop	pop ecx
	loop display_loop
mov ecx, 5	
mov esi, 0	print newline, nlen
mov ebx, 0	
sum_loop:	print msg2, slen
add ebx, [array + esi*4]	mov eax, [sum]
add esi, 1	mov ecx, dispbuf
loop sum_loop	add ecx, 5
mov [sum], ebx	mov byte [ecx], 0
print msg1, mlen	mov ebx, 10
mov ecx, 5	convert_sum:
mov esi, 0	dec ecx
display_loop:	xor edx, edx
push ecx	div ebx
mov eax, [array + esi*4]	add dl, '0'
	mov [ecx], dl
mov ecx, dispbuf	test eax, eax
add ecx, 5	jnz convert_sum
mov byte [ecx], 0	
mov ebx, 10	print ecx, 6
convert_to_ascii:	print newline, nlen
dec ecx	
xor edx, edx	mov eax, 1
div ebx	mov ebx, 0
add dl, '0'	int 80h

### OUTPUT –

```
root@Atharv:/mnt/c/Users/Athar/OneDrive/Documents/college/SEM4/MPMC/Labs/exp8# ./5
Enter the elements of the array:
56
45
12
102
78
The array elements are: 56 45 12 102 78
Sum of elements: 293
```

### CONCLUSION –

Array operations for inputting ,display and counting were successfully implemented using NASM .