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'
    mov [num], al
    print num, 1
    cmp ecx, 1
    je skip_space
    print space, slen
skip_space:
    inc esi
    pop ecx
    loop display_loop

    print newline, nlen

    mov eax, 1
    mov ebx, 0
    int 80h
```

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
   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 (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 equu $ - newline

section .bss
   array resb 5
   num resb 3
   pos_count resb 1
   neg_count resb 1

section .text
   global _start

_start:
   mov ecx, 5
   mov esi, 0
   mov byte[pos_count], 0
   mov byte[neg_count], 0

   push ecx
   print msg, plen
   print newline, nlen
   pop ecx

input_loop:
   push ecx
   read num, 3
   mov al, [num]
   cmp al, '-'
   je negative
   sub al, '0'
   inc byte[pos_count]
   jmp store
negative:
   mov al, [num + 1]
   sub al, '0'
```

```
    neg al
    inc byte[neg_count]
store:
    mov [array + esi], al
    inc esi
    pop ecx
    loop input_loop

    print pos_msg, pos_len
    mov al, [pos_count]
    add al, '0'
    mov [num], al
    print num, 1
    print newline, nlen

    print neg_msg, neg_len
    mov al, [neg_count]
    add al, '0'
    mov [num], al
    print num, 1
    print newline, nlen

    mov eax, 1
    mov ebx, 0
    int 80h
```

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]
    mov ah, 0
    mov bl, 2
    div bl
    cmp ah, 0
    jz even_number
    inc byte[odd_count]
    jmp next_number
even_number:
    inc byte[even_count]
next_number:
    inc esi
    loop count_loop

    print msg1, mlen1
    mov al, [even_count]
    add al, '0'
    mov [num], al
    print num, 1
    print newline, nlen

    print msg2, mlen2
    mov al, [odd_count]
    add al, '0'
    mov [num], al
    print num, 1
    print newline, nlen

    mov eax, 1
    mov ebx, 0
    int 80h
```

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
```

4] Write an ALP to calculate elements less than 5 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 '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
   array resb 5
   num resb 2
   countLess resb 1
   countGreater resb 1
   countEqual resb 1

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

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

```
    mov byte[countGreater], 0
    mov byte[countEqual], 0

count_loop:
    mov al, [array + esi]
    cmp al, 5
    je equal_to_5
    jg greater_than_5
    inc byte[countLess]
    jmp continue_count
greater_than_5:
    inc byte[countGreater]
    jmp continue_count
equal_to_5:
    inc byte[countEqual]
continue_count:
    inc esi
    loop count_loop

    print msg1, mlen
    mov al, [countLess]
    add al, '0'
    mov [num], al
    print num, 1
    print newline, nlen

    print msg2, m2len
    mov al, [countGreater]
    add al, '0'
    mov [num], al
    print num, 1
    print newline, nlen

    print msg3, m3len
    mov al, [countEqual]
    add al, '0'
    mov [num], al
    print num, 1
    print newline, nlen

    mov eax, 1
    mov ebx, 0
    int 80h
```

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
```

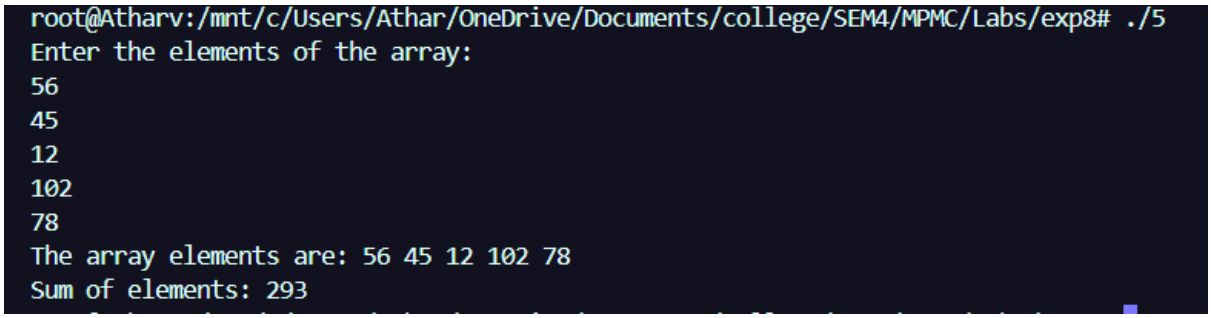
5] Write an ALP to find the sum of elements of the array

INPUT –

```
%macro print 2
   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
   msg2 db 'Sum of elements: '
   slen equ $ - msg2
   space db ' '
   splen equ $ - space
   newline db 0xA
   nlen equ $ - newline

section .bss
   array resd 5
   num resb 6
   sum resd 1
   dispbuf resb 6

section .text
   global _start

_start:
   mov ecx, 5
   mov esi, 0

   push ecx
   print msg, plen
   pop ecx

input_loop:
   push ecx
   read num, 6

   mov ebx, 0
   mov ecx, 0
convert:
   mov al, [num + ecx]
   cmp al, 0xA
   je done_convert
   sub al, '0'
   imul ebx, 10
   add bl, al
```

```
    inc ecx
    jmp convert

done_convert:
    mov [array + esi*4], ebx
    add esi, 1
    pop ecx
    loop input_loop

    mov ecx, 5
    mov esi, 0
    mov ebx, 0
sum_loop:
    add ebx, [array + esi*4]
    add esi, 1
    loop sum_loop
    mov [sum], ebx
    print msg1, mlen
    mov ecx, 5
    mov esi, 0
display_loop:
    push ecx
    mov eax, [array + esi*4]

    mov ecx, dispbuf
    add ecx, 5
    mov byte [ecx], 0
    mov ebx, 10
convert_to_ascii:
    dec ecx
    xor edx, edx
    div ebx
    add dl, '0'
    mov [ecx], dl
    test eax, eax
    jnz convert_to_ascii

    print ecx, 6
    print space, splen
    add esi, 1
    pop ecx
    loop display_loop

    print newline, nlen

    print msg2, slen
    mov eax, [sum]
    mov ecx, dispbuf
    add ecx, 5
    mov byte [ecx], 0
    mov ebx, 10
convert_sum:
    dec ecx
    xor edx, edx
    div ebx
    add dl, '0'
    mov [ecx], dl
    test eax, eax
    jnz convert_sum

    print ecx, 6
    print newline, nlen

    mov eax, 1
    mov ebx, 0
    int 80h
```

OUTPUT –



CONCLUSION –

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