**1]Write an ALP to implement write system to display a number using macros**

**Program –**

section .data

    outputMsg db "You entered: ", 0

    newline db 0xa

    %macro print 2

        mov eax, 4

        mov ebx, 1

        mov ecx, %1

        mov edx, %2

        int 0x80

    %endmacro

number db '5'

section .text

    global \_start

\_start:

    print outputMsg, 16

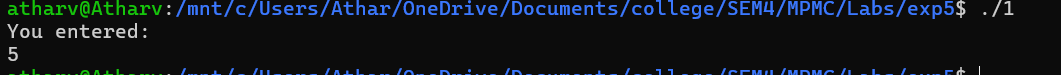
    print newline, 1

    mov eax, 1

    mov ebx, 0

    int 0x80

**OUTPUT –**

****

**2] Write an ALP to implement write system to display two input numbers using macros**

**Program -**

section .data

    msg1 db "Enter first number : ", 0xa

    msg2 db "Enter second number : ", 0xa

    result\_msg db "Your printed number is : ", 0xa

    msg3 db "First number is: ", 0xa

    msg4 db "Second number is: ", 0xa

    newline db 0xa

    space db " "

section .bss

    num1 resb 2

    num2 resb 2

%macro print 2

    mov eax, 4

    mov ebx, 1

    mov ecx, %1

    mov edx, %2

    int 0x80

%endmacro

%macro read 2

    mov eax, 3

    mov ebx, 0

    mov ecx, %1

    mov edx, %2

    int 0x80

%endmacro

section .text

    global \_start

\_start:

    print msg1, 19

    read num1, 2

    print space, 1

    print result\_msg, 22

    print space, 1

    print num1, 2

    print msg2, 19

    print space, 1

    read num2, 2

    print result\_msg, 22

    print space, 1

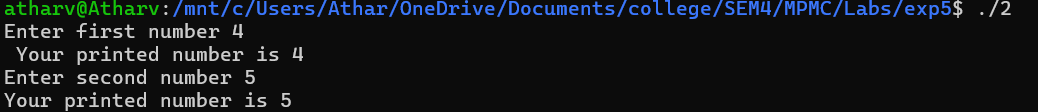
    print num2, 2

    mov eax, 1

    mov ebx, 0

    int 0x80

**OUTPUT**

****

**3]Write an ALP to implement write system call using macros**

**Program –**

section .data

    msg db "Enter your number : ", 0

    outputMsg db "You entered: ", 0

    newline db 0xa

section .bss

    number resb 1

%macro print 2

    mov eax, 4

    mov ebx, 1

    mov ecx, %1

    mov edx, %2

    int 0x80

%endmacro

%macro read 2

    mov eax, 3

    mov ebx, 0

    mov ecx, %1

    mov edx, %2

    int 0x80

%endmacro

section .text

    global \_start

\_start:

    print msg, 20

    read number, 1

    print outputMsg, 16

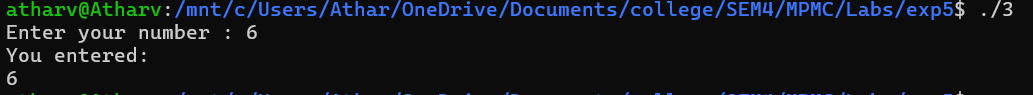
    print newline, 1

    mov eax, 1

    mov ebx, 0

    int 0x80

**Output –**

****

**4]Write an ALP to implement calculator functions using macros**

**Program –**

section .data

msg db ' ',10

msgLen equ $-msg

msg1 db 'Number 1: '

msg1Len equ $-msg1

msg2 db 'Number 2: '

msg2Len equ $-msg2

msg3 db 'Sum: '

msg3Len equ $-msg3

msg4 db 'Difference: '

msg4Len equ $-msg4

msg5 db 'Product: '

msg5Len equ $-msg5

msg6 db 'Quotient: '

msg6Len equ $-msg6

msg7 db 'Remainder: '

msg7Len equ $-msg7

%macro writesystem 2

mov eax,4

mov ebx,1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

%macro readsystem 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

%endmacro

%macro addition 2

mov eax, [num1]

sub eax, '0'

mov ebx, [num2]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [sum], eax

%endmacro

%macro subtraction 2

mov eax, [num1]

sub eax, '0'

mov ebx, [num2]

sub ebx, '0'

sub eax, ebx

add eax, '0'

mov [diff], eax

%endmacro

%macro multiplication 2

mov eax, [num1]

sub eax, '0'

mov ebx, [num2]

sub ebx, '0'

mul ebx

add eax, '0'

mov [prod], eax

%endmacro

%macro division 2

mov al, [num1]

sub al, '0'

mov bl, [num2]

sub bl, '0'

div bl

add al, '0'

mov [quot], al

add ah, '0'

mov [rem], ah

%endmacro

section .bss

num1 RESB 5

num2 RESB 5

sum RESB 5

diff RESB 5

prod RESB 5

quot RESB 5

rem RESB 5

section .text

global \_start

\_start:

writesystem msg1,msg1Len

readsystem num1,5

writesystem msg2,msg2Len

readsystem num2,5

addition num1,num2

writesystem msg3,msg3Len

writesystem sum,1

writesystem msg, msgLen

subtraction num1,num2

writesystem msg4,msg4Len

writesystem diff,1

writesystem msg, msgLen

multiplication num1,num2

writesystem msg5,msg5Len

writesystem prod, 1

writesystem msg, msgLen

division num1,num2

writesystem msg6,msg6Len

writesystem quot, 1

writesystem msg, msgLen

writesystem msg7,msg7Len

writesystem rem, 1

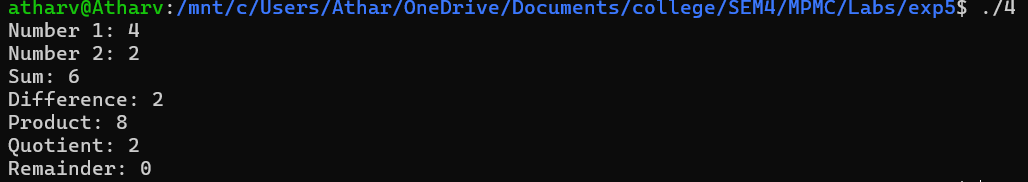
writesystem msg, msgLen

mov eax, 1

mov ebx, 0

int 80h

**OUTPUT –**

****

**5]Write an ALP to print the Fibonacci series till n terms using macros**

**Program –**

%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, 0

    mov ecx, %1

    mov edx, %2

    int 80h

%endmacro

%macro ADD 2

    movzx eax, byte [%1]

    sub al, '0'

    movzx ebx, byte [%2]

    sub bl, '0'

    add eax, ebx

    add al, '0'

    mov [result], al

%endmacro

section .data

    prompt db 'Enter n: '

    prompt\_len equ $ - prompt

    msg db 'Series: '

    msg\_len equ $ - msg

    space db ' '

    newline db 10

section .bss

    n resb 2

    num1 resb 2

    num2 resb 2

    result resb 2

section .text

global \_start

\_start:

    write prompt, prompt\_len

    read n, 2

    write msg, msg\_len

    mov byte [num1], '0'

    mov byte [num2], '1'

    movzx ecx, byte [n]

    sub ecx, '0'

    loop:

    push ecx

    write num1, 1

    write space, 1

    ADD num1, num2

    mov al, [num2]

    mov [num1], al

    mov al, [result]

    mov [num2], al

    pop ecx

    dec ecx

    jnz loop

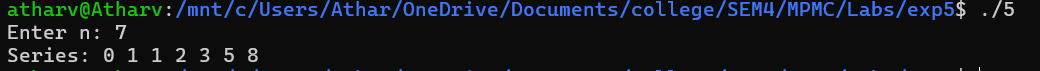
    write newline, 1

mov eax, 1

mov ebx,0

int 80h

**OUTPUT –**

****

**6]Write an ALP to print your name 7 times using macros**

**Program –**

%macro writenumber 2

    mov ecx, %1

    mov edx, %2

    mov ebx, 1

    mov eax, 4

    int 80h

%endmacro

section .data

    msg db "Atharv", 010

    len equ $ - msg

section .text

global \_start

\_start:

    mov esi, 0

    loop:

    push esi

    writenumber msg, len

    pop esi

    inc esi

    cmp esi, 7

    jl loop

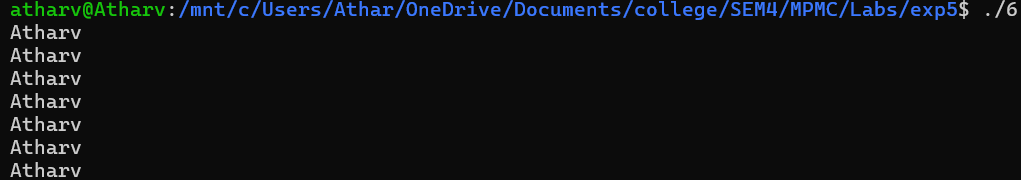
end:

mov eax, 1

mov ebx,0

int 80h

**OUTPUT –**

****

**7] Write an ALP to implement to take two inputs from user using macros**

**Program –**

section .data

    message1 db "Enter first value: ", 0

    message2 db "Enter second value: ", 0

    output1 db "First value: ", 0

    output2 db "Second value: ", 0

    newline db 0xa

    space db " "

section .bss

    value1 resb 32

    value2 resb 32

%macro print 2

    mov eax, 4

    mov ebx, 1

    mov ecx, %1

    mov edx, %2

    int 0x80

%endmacro

%macro read 2

    mov eax, 3

    mov ebx, 0

    mov ecx, %1

    mov edx, %2

    int 0x80

%endmacro

%macro exit 0

    mov eax, 1

    mov ebx, 0

    int 0x80

%endmacro

section .text

    global \_start

\_start:

    print message1, 17

    print space, 1

    read value1, 32

    print message2, 18

    print space, 1

    read value2, 32

    print output1, 12

    print space, 1

    print value1, 32

    print newline, 1

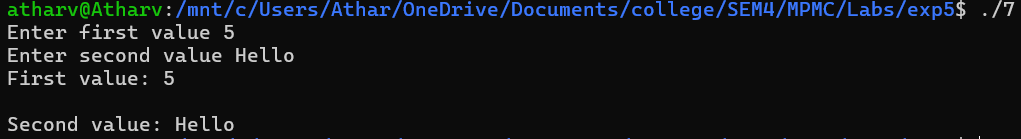
    print output2, 13

    print space, 1

    print value2, 32

    exit

**OUTPUT –**

****

**Conclusion – Macros were successfully implemented using NASM and UBUNTU**