**5A]** Write an assembly language program to implement the write system call to display a number

using macros.

**Program:**

section .data

string db 'The number is: ', 10

stringlen equ $-string

newline db '',10

newlinelen equ $-newline

num equ 4

section .bss

buffer resb 1

section .text

global \_start

%macro write 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

%macro stringconv 1

mov eax, %1

mov ebx, 10

mov edx, 0

div ebx

add edx, '0'

mov byte [buffer], dl

mov esi, buffer

%endmacro

\_start:

write string,stringlen

stringconv num

write buffer,1

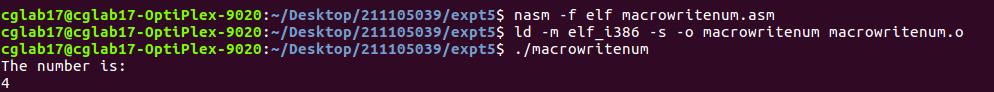
write newline,newlinelen

mov eax,1

xor ebx,ebx

int 0x80

**Output:**



**5B]** Write an assembly language program to implement the write system call to display two

inputs using macros.

**Program:**

section .data

string1 db 'Hello There',10

string1len equ $-string1

string2 db 'General Kenobi',10

string2len equ $-string2

output1 db 'String 1 is:',10

output1len equ $-output1

output2 db 'String 2 is:',10

output2len equ $-output2

newline db '',10

newlinelen equ $-newline

section .text

global \_start

%macro write 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

\_start:

write output1,output1len

write string1,string1len

write newline,newlinelen

write output2,output2len

write string2,string2len

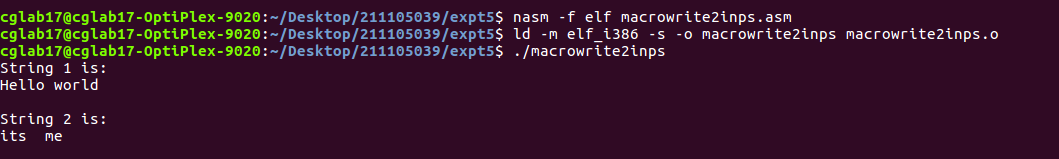
write newline,newlinelen

mov eax,1

xor ebx,ebx

int 0x80

**Output:**



**5C]** Write an assembly language program to implement the read system call using macros to

accept a number and display it.

**Program:**

section .data

string1 db 'Enter a number:',10

string1len equ $-string1

string2 db 'The number is: ', 10

string2len equ $-string2

newline db '',10

newlinelen equ $-newline

section .bss

num resb 4

section .text

global \_start

%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

\_start:

write string1,string1len

read num,4

write newline,newlinelen

write string2,string2len

write num,4

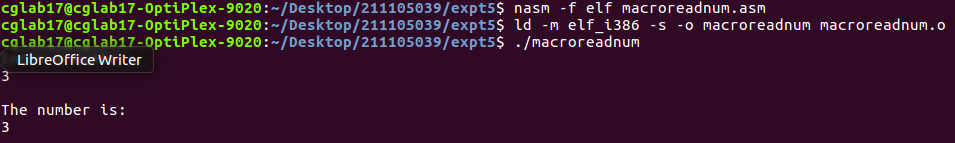
write newline,newlinelen

mov eax,1

xor ebx,ebx

int 0x80

**Output:**



**5D]** Write an assembly language program to implement the read system call using macros to

accept two inputs and display it.

**Program:**

section .data

string1 db 'Enter string 1:',10

string1len equ $-string1

string2 db 'Enter string 2:',10

string2len equ $-string2

output1 db 'String 1 is:',10

output1len equ $-output1

output2 db 'String 2 is:',10

output2len equ $-output2

newline db '',10

newlinelen equ $-newline

section .bss

str1 resb 10

str2 resb 10

section .text

global \_start

%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

\_start:

write string1,string1len

read str1,10

write newline,newlinelen

write string2,string1len

read str2,10

write newline,newlinelen

write output1,output1len

write str1,10

write newline,newlinelen

write output2,output2len

write str2,10

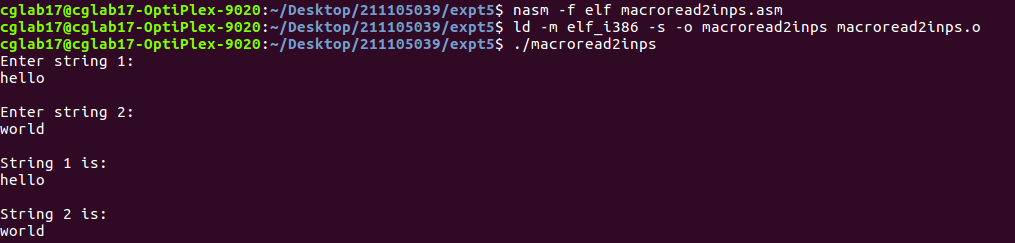
write newline,newlinelen

mov eax,1

xor ebx,ebx

int 0x80

**Output:**



**5E]** Write an assembly language program to implement calculator program using macros.

Program:

%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

%macro newline 0

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 0x80

%endmacro

%macro SUM 3

mov al, [%1]

sub al, '0'

mov bl, [%2]

sub bl, '0'

add al, bl

add al, '0'

mov [%3], al

int 0x80

%endmacro

%macro DIFF 3

mov eax, [%1]

sub eax, '0'

mov ebx, [%2]

sub ebx, '0'

sub eax, ebx

add eax, '0'

mov [%3], eax

%endmacro

%macro PROD 3

mov al, [%1]

sub al, '0'

mov bl, [%2]

sub bl, '0'

mul bl

add al, '0'

mov [%3], al

%endmacro

%macro DIV 4

mov al, [%1]

sub al, '0'

mov bl, [%2]

sub bl, '0'

div bl

add al, '0'

mov [%3], al

add ah, '0'

mov [%4], ah

%endmacro

section .data

str1 db "Enter the numbers: "

str1len equ $-str1

read num1, num2

sumstr db "Sum is: "

sumlen equ $-sumstr

diffstr db "Diff is: "

difflen equ $-diffstr

prodstr db "Prod is: "

prodlen equ $-prodstr

quostr db "Quotient is: "

quolen equ $-quostr

remstr db "Remainder is: "

remlen equ $-remstr

nl db "", 10

nllen equ $-nl

section .bss

num1 resb 4

num2 resb 4

prod resb 4

quo resb 4

rem resb 4

sum resb 4

diff resb 4

section .text

global \_start

\_start:

write str1, str1len

read num1, 4

read num2, 4

SUM num1, num2, sum

write sumstr, sumlen

write sum, 4

newline

DIFF num1, num2, diff

write diffstr, difflen

write diff, 4

newline

PROD num1, num2, prod

write prodstr, prodlen

write prod, 4

newline

DIV num1, num2, quo, rem

write quostr, quolen

write quo, 4

newline

write remstr, remlen

write rem, 4

newline

mov eax,1

xor ebx,ebx

int 0x80

**Output:**



**5F]** Write an assembly language program to print the Fibonacci series till n.

**Program:**

%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

%macro newline 0

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 0x80

%endmacro

%macro form 3

mov eax, [%1]

sub eax, '0'

mov ebx, [%2]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [%3], eax

int 0x80

%endmacro

section .bss

n resb 4

a resb 4

b resb 4

c resb 4

i resb 4

section .data

str1 db 'Enter a number: '

str1len equ $-str1

str2 db 'Fibonacci series: '

str2len equ $-str2

nl db "", 10

nllen equ $-nl

spa db " "

spalen equ $-spa

section .text

global \_start

\_start:

write str1, str1len

read n, 4

write str2, str2len

mov byte[i], '0'

mov byte[a], '0'

mov byte[b], '1'

cmp byte[n], '0'

JE L4

JMP L1

L1:

write a, 4

write spa,spalen

inc byte[i]

mov al, [i]

cmp al, byte[n]

JE L4

JMP L2

L2:

write b, 4

write spa,spalen

inc byte[i]

mov al, [i]

cmp al, byte[n]

JE L4

JMP L3

L3:

form a, b, c

write c, 4

write spa,spalen

mov al, [b]

mov [a], al

mov al, [c]

mov [b], al

inc byte[i]

mov al, [i]

cmp al, byte[n]

JE L4

JMP L3

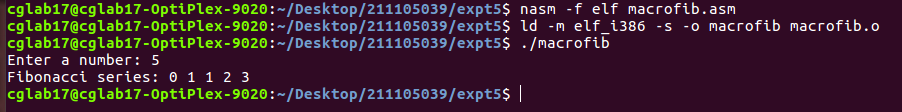
L4:

newline

mov eax, 1

int 80h

**Output:**



**5G]** Write an assembly language program to print your name 9 times.

**Program:**

%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

%macro newline 0

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 0x80

%endmacro

section .data

str1 db "Enter your name: "

len1 equ $-str1

nl db "",10

nllen equ $-nl

spa db " "

spalen equ $-spa

section .bss

name resw 15

count resb 4

section .text

global \_start

\_start:

write str1,len1

read name,15

mov byte[count],'1'

L1:

write count,4

write spa,spalen

write name,15

inc byte[count]

cmp byte[count],'9'

JLE L1

mov eax,1

xor ebx,ebx

int 0x80

**Output:**

