a. Write an assembly language program to implement the write system call to display a

number using macros.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

%macro Write\_number 1

mov eax, write

mov ebx, stdout

mov ecx, %1

mov edx, 1

int 80h

%endmacro

section .data

prompt1 db "Enter a numbers : ",10

prompt1len equ $-prompt1

prompt2 db "Entered number: ",8

prompt2len equ $-prompt2

newline db 10,0

section .bss

num1: resb 1

section .text

global \_start

\_start:

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

mov eax, read

mov ebx, stdin

mov ecx, num1

mov edx, 2

int 80h

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

Write\_number num1

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

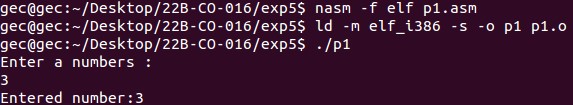
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**



b. Write an assembly language program to implement the write system call to display

two inputs using macros

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

%macro Write\_numbers 4

mov eax, write

mov ebx, stdout

mov ecx, %3

mov edx, %4

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %1

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %2

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

%endmacro

section .data

prompt1 db "Enter two numbers : ",10

prompt1len equ $-prompt1

prompt2 db "Entered numbers are : ",10

prompt2len equ $-prompt2

newline db 10,0

section .bss

num1: resb 1

num2: resb 1

section .text

global \_start

\_start:

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

mov eax, read

mov ebx, stdin

mov ecx, num1

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, num2

mov edx, 2

int 80h

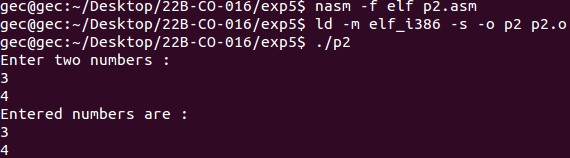
Write\_numbers num1, num2, prompt1, prompt1len

mov eax, exit

mov ebx, 0

int 80h

**Output:**



c. Write an assembly language program to implement the read system call using

macros to accept a number and display it.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

%macro Read\_and\_Display\_number 3

mov eax, read

mov ebx, stdin

mov ecx, %1

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %2

mov edx, %3

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %1

mov edx, 1

int 80h

%endmacro

section .data

prompt1 db "Enter a numbers : ",8

prompt1len equ $-prompt1

prompt2 db "Entered number: ",8

prompt2len equ $-prompt2

newline db 10,0

section .bss

num1: resb 1

section .text

global \_start

\_start:

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

Read\_and\_Display\_number num1, prompt2, prompt2len

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

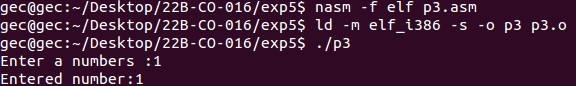
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**



d. Write an assembly language program to implement the read system call using

macros to accept two inputs and display it.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

%macro Read\_and\_Display\_two\_number 4

mov eax, read

mov ebx, stdin

mov ecx, %1

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, %2

mov edx, 2

int 80h

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %1

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %2

mov edx, 1

int 80h

%endmacro

section .data

prompt1 db "Enter 2 numbers : ",10

prompt1len equ $-prompt1

prompt2 db "Entered numbers are : ",10

prompt2len equ $-prompt2

newline db 10,0

section .bss

num1: resb 1

num2: resb 1

section .text

global \_start

\_start:

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

Read\_and\_Display\_two\_number num1, num2, prompt2, prompt2len

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

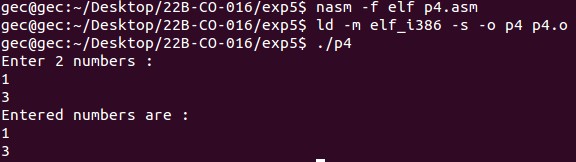
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**



e. Write an assembly language program to implement calculator program using

macros.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

; Add

%macro ADDITION 3

mov eax, [%1]

sub eax, '0'

mov ebx, [%2]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [%3], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %3

mov edx, 1

int 80h

%endmacro

; sub

%macro SUBTRACTION 3

mov eax, [%1]

sub eax, '0'

mov ebx, [%2]

sub ebx, '0'

sub eax, ebx

add eax, '0'

mov [%3], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %3

mov edx, 1

int 80h

%endmacro

; Mul

%macro MULTIPLICATION 3

mov ax, [%1]

sub ax, '0'

mov bx, [%2]

sub bx, '0'

mul bx

add ax, '0'

mov [%3], ax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, %3

mov edx, 1

int 80h

%endmacro

; Div

%macro DIVISION 4

mov al, [%1]

sub al, '0'

mov bl, [%2]

sub bl, '0'

div bl

add al, '0'

add ah, '0'

mov [%3], ah

mov [%4], al

int 80h

%endmacro

section .data

prompt1 db "Enter two numbers : ",10

prompt1len equ $-prompt1

prompt2 db "Sum of numbers : ",8

prompt2len equ $-prompt2

prompt3 db "Subtraction of numbers : ", 8

prompt3len equ $-prompt3

prompt4 db "Produt of numbers : ", 8

prompt4len equ $-prompt4

prompt5 db "Division of numbers : ", 10

prompt5len equ $-prompt5

prompt6 db "quot : ", 8

prompt6len equ $-prompt6

prompt7 db "rmd : ",8

prompt7len equ $-prompt7

newline db 10,0

section .bss

num1: resb 1

num2: resb 2

result: resb 1

remd: resb 1

quot: resb 1

section .text

global \_start

\_start:

;Inputs

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

mov eax, read

mov ebx, stdin

mov ecx, num1

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, num2

mov edx, 2

int 80h

;adds

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

ADDITION num1, num2, result

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

;subs

mov eax, write

mov ebx, stdout

mov ecx, prompt3

mov edx, prompt3len

int 80h

SUBTRACTION num1, num2, result

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

;multi's

mov eax, write

mov ebx, stdout

mov ecx, prompt4

mov edx, prompt4len

int 80h

MULTIPLICATION num1, num2, result

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

;div's

mov eax, write

mov ebx, stdout

mov ecx, prompt5

mov edx, prompt5len

int 80h

DIVISION num1, num2, remd, quot

mov eax, write

mov ebx, stdout

mov ecx, prompt6

mov edx, prompt6len

int 80h

mov eax, write

mov ebx, stdout

mov ecx, quot

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, prompt7

mov edx, prompt7len

int 80h

mov eax, write

mov ebx, stdout

mov ecx, remd

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

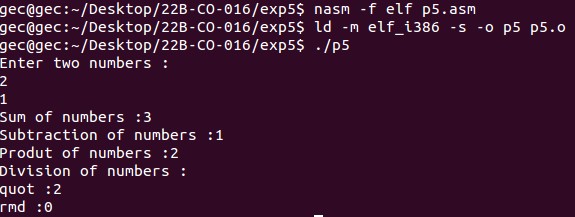
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**



f. Write an assembly language program to print the fibonacci series till n.

%macro Read 2

mov eax, 3

mov ebx, 0

mov ecx, %1

mov edx, %2

int 80h

%endmacro

%macro Write 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

%macro Fibonacci 3

mov al, [%1]

mov bl, [%2]

sub al, '0'

sub bl, '0'

add al, bl

add al, '0'

mov [%3], al

int 80h

%endmacro

section .data

prompt1 db "Enter number of elements of fibonacci : ",8

prompt1len equ $-prompt1

prompt2 db "Fibonacci : ",10

prompt2len equ $-prompt2

newline db 10,0

a db '0',0

b db '1',0

c db '0',0

count db '0',0

section .bss

n: resb 1

section .text

global \_start

\_start:

Write prompt1, prompt1len

Read n, 2

mov al, [count]

mov bl, [n]

cmp al, bl

je exit

Write prompt2, prompt2len

Write a, 1

Write newline, 1

inc byte [count]

mov al, [count]

mov bl, [n]

cmp al, bl

je exit

Write b, 1

Write newline, 1

inc byte [count]

mov al, [count]

mov bl, [n]

cmp al, bl

je exit

jmp loop

loop:

Fibonacci a,b,c

Write c,1

Write newline,1

mov bl, [b]

mov [a], bl

mov bl, [c]

mov [b], bl

inc byte [count]

mov al, [count]

mov bl, [n]

cmp al, bl

je exit

jmp loop

exit:

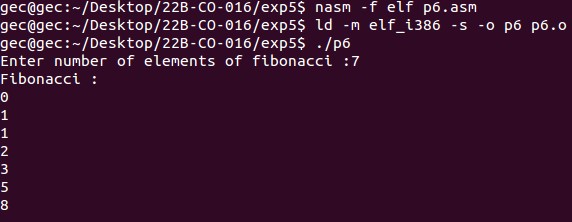
Write newline,1

mov eax, 1

mov ebx, 0

int 80h

**Output:**



g. Write an assembly language program to print your name 9 times.

%MACRO write\_string 2

mov eax, 4

mov ebx, 1

push ebx

mov ecx ,%1

mov edx, %2

int 80h

pop ecx

%ENDMACRO

%MACRO read\_string 2

mov eax, 3

mov ebx, 2

mov ecx, %1

mov edx, %2

int 80h

%ENDMACRO

section .data

s1 db 'Enter the name: '

s1l equ $-s1

s2 db 'looping the name: ', 10

s2l equ $-s2

section .bss

name resb 40

section .text

global \_start

\_start:

write\_string s1, s1l

read\_string name, 40

write\_string s2, s2l

mov ecx, 9

loop\_name:

write\_string name, 40

loop loop\_name

mov eax, 1

mov ebx, 0

int 80h

