a. Write an assembly language program to enter two numbers and display their sum on screen using EQU directive.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

section .data

prompt1 db "Enter two numbers : ",10

prompt1len equ $-prompt1

prompt2 db "sum of numbers : ",8

prompt2len equ $-prompt2

newline db 10,0

section .bss

num1: resb 1

num2: resb 1

result: 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, 5

int 80h

mov eax, read

mov ebx, stdin

mov ecx, num2

mov edx, 5

int 80h

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

mov eax, [num1]

sub eax, '0'

mov ebx, [num2]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [result], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

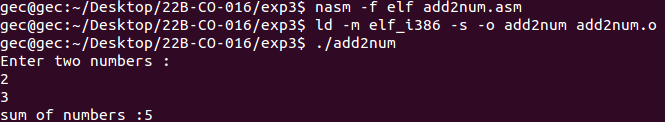
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**



b. Write an assembly language program to perform addition, subtraction, multiplication and division of two entered numbers.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

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

n1: resb 1

n2: 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, n1

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, n2

mov edx, 2

int 80h

; Add

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

mov eax, [n1]

sub eax, '0'

mov ebx, [n2]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [result], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

; sub

mov eax, write

mov ebx, stdout

mov ecx, prompt3

mov edx, prompt3len

int 80h

mov eax, [n1]

sub eax, '0'

mov ebx, [n2]

sub ebx, '0'

sub eax, ebx

add eax, '0'

mov [result], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

; Mul

mov eax, write

mov ebx, stdout

mov ecx, prompt4

mov edx, prompt4len

int 80h

mov ax, [n1]

sub ax, '0'

mov bx, [n2]

sub bx, '0'

mul bx

add ax, '0'

mov [result], ax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

; Div

mov eax, write

mov ebx, stdout

mov ecx, prompt5

mov edx, prompt5len

int 80h

mov al, [n1]

sub al, '0'

mov bl, [n2]

sub bl, '0'

mul bl

add al, '0'

add ah, '0'

mov [remd], ah

mov [quot], al

int 80h

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, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

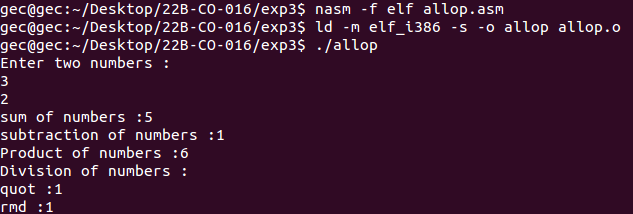
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**

****

c. Write an assembly language program to find area and perimeter of rectangle and triangle.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

section .data

prompt1 db "Enter both sides of rectangle : ",10

prompt1len equ $-prompt1

prompt2 db "Enter base and height of triangle : ",10

prompt2len equ $-prompt2

prompt3 db "Area of rectangle : ", 8

prompt3len equ $-prompt3

prompt4 db "Area of triangle : ", 8

prompt4len equ $-prompt4

prompt5 db "Parameter of rectangle : ", 8

prompt5len equ $-prompt5

prompt6 db "Parameter of triangle : ", 8

prompt6len equ $-prompt6

prompt7 db "Enter all three side of the triangle : ",10

prompt7len equ $-prompt7

newline db 10,0

section .bss

s1: resb 1

s2: resb 1

s3: resb 1

area: resb 1

par: resb 1

section .text

global \_start

\_start:

;rectangle inputs

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

mov eax, read

mov ebx, stdin

mov ecx, s1

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, s2

mov edx, 2

int 80h

;area of rectangle

mov eax, write

mov ebx, stdout

mov ecx, prompt3

mov edx, prompt3len

int 80h

mov al, [s1]

sub al, '0'

mov bl, [s2]

sub bl, '0'

mul bl

add ax, '0'

mov [area], ax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, area

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

; Parameter of rectangle

mov eax, write

mov ebx, stdout

mov ecx, prompt5

mov edx, prompt5len

int 80h

mov eax, [s1]

sub eax, '0'

mov ebx, [s2]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [par], eax

int 80h

mov al, [par]

sub al, '0'

mov bl, "2"

sub bl, '0'

mul bl

add ax, '0'

mov [par], ax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, par

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

;triangle inputs

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

mov eax, read

mov ebx, stdin

mov ecx, s1

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, s2

mov edx, 2

int 80h

;area of triangle

mov eax, write

mov ebx, stdout

mov ecx, prompt4

mov edx, prompt4len

int 80h

mov ax, [s1]

sub ax, '0'

mov bx, [s2]

sub bx, '0'

mul bx

add ax, '0'

mov [area], ax

int 80h

mov dx, 0

mov ax, [area]

sub ax, '0'

mov bx, "2"

sub bx, '0'

div bx

add ax, '0'

mov [area], ax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, area

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

int 80h

; Parameter of triangle

mov eax, write

mov ebx, stdout

mov ecx, prompt7

mov edx, prompt7len

int 80h

mov eax, read

mov ebx, stdin

mov ecx, s1

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, s2

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, s3

mov edx, 2

int 80h

mov eax, write

mov ebx, stdout

mov ecx, prompt6

mov edx, prompt6len

int 80h

mov eax, [s1]

sub eax, '0'

mov ebx, [s2]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [par], eax

int 80h

mov eax, [s3]

sub eax, '0'

mov ebx, [par]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [par], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, par

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, newline

mov edx, 1

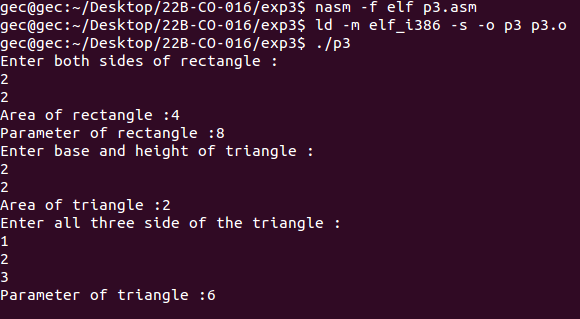
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**



d. Write an assembly language program to display sum of 3 entered numbers.

exit equ 1

close equ 0

read equ 3

write equ 4

stdin equ 0

stdout equ 1

section .data

prompt1 db "Enter 3 numbers ",10

prompt1len equ $-prompt1

prompt2 db "Sum of 3 numbers entered is : ",8

prompt2len equ $-prompt2

newline db 10,0

section .bss

a: resb 1

b: resb 1

c: 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, a

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, b

mov edx, 2

int 80h

mov eax, read

mov ebx, stdin

mov ecx, c

mov edx, 2

int 80h

;add

mov eax, [a]

sub eax, '0'

mov ebx, [b]

sub ebx, '0'

add eax, ebx

mov ebx, [c]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [a], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

mov eax, write

mov ebx, stdout

mov ecx, a

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

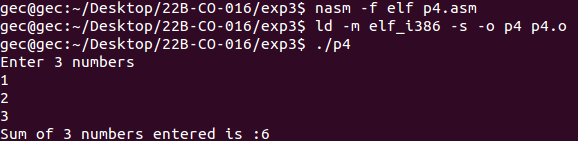
int 80h

mov eax, exit

mov ebx, close

int 80h

**Output:**



e. Write an assembly language program that inputs a number and display the number and the next 4 numbers using increment operation.

exit equ 1

close equ 0

read equ 3

write equ 4

stdin equ 0

stdout equ 1

section .data

prompt1 db "Enter a number ",10

prompt1len equ $-prompt1

prompt2 db "Entered Number with next 4 consiqutive numbers : ",10

prompt2len equ $-prompt2

comma db ",",0

newline db 10,0

section .bss

a: 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, a

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, a

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

; next process

mov eax, [a]

sub eax, '0'

mov ebx, '1'

add eax, ebx

mov [a], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, a

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

mov eax, [a]

sub eax, '0'

mov ebx, '1'

add eax, ebx

mov [a], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, a

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

mov eax, [a]

sub eax, '0'

mov ebx, '1'

add eax, ebx

mov [a], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, a

mov edx, 1

int 80h

mov eax, [a]

sub eax, '0'

mov ebx, '1'

add eax, ebx

mov [a], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, a

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

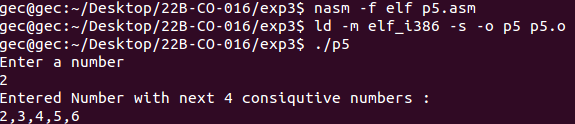
int 80h

mov eax, exit

mov ebx, close

int 80h

**Output:**

****

f. Write an assembly language program to calculate the area of the circle.

exit equ 1

read equ 3

write equ 4

stdin equ 0

stdout equ 1

section .data

prompt1 db "Enter radius of circle : ",10

prompt1len equ $-prompt1

prompt2 db "Area of circle : ",10

prompt2len equ $-prompt2

newline db 10, 0

section .bss

r: resb 1

area: resb 1

section .text

global \_start

\_start:

;circle inputs

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

mov eax, read

mov ebx, stdin

mov ecx, r

mov edx, 2

int 80h

;area

mov eax, write

mov ebx, stdout

mov ecx, prompt2

mov edx, prompt2len

int 80h

mov ax, [r]

sub ax, "0"

mov bx, [r]

sub bx, "0"

mul bx

add ax, "0"

mov [area], ax

int 80h

mov ax, [area]

sub ax, "0"

mov bx, "3"

sub bx, "0"

mul bx

add ax, "0"

mov [area], ax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, area

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

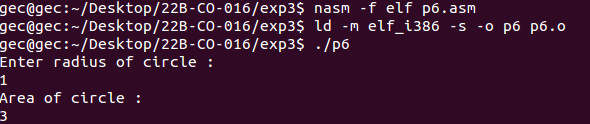
int 80h

mov eax, exit

mov ebx, 0

int 80h

**Output:**



g. Write an assembly language program to display all odd numbers from 0 to 9.

exit equ 1

close equ 0

read equ 3

write equ 4

stdin equ 0

stdout equ 1

section .data

prompt1 db "Odd numebrs from 0 - 9 :",10

prompt1len equ $-prompt1

odd db "1",0

inct db "2",0

comma db ",",0

newline db 10,0

section .bss

result: resb 1

section .text

global \_start

\_start:

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

;1

mov eax, write

mov ebx, stdout

mov ecx, odd

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

;3

mov eax, [odd]

sub eax, '0'

mov ebx, [inct]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [result], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

;5

mov eax, [result]

sub eax, '0'

mov ebx, [inct]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [result], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

;7

mov eax, [result]

sub eax, '0'

mov ebx, [inct]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [result], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

;9

mov eax, [result]

sub eax, '0'

mov ebx, [inct]

sub ebx, '0'

add eax, ebx

add eax, '0'

mov [result], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

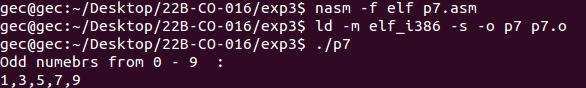
int 80h

mov eax, exit

mov ebx, close

int 80h

**Output:**



h. Write an assembly language program to find the multiples of 3 from 0 to 9.

exit equ 1

close equ 0

read equ 3

write equ 4

stdin equ 0

stdout equ 1

section .data

prompt1 db "Multiple of 3 from 0 - 9 :",10

prompt1len equ $-prompt1

comma db ",",0

newline db 10,0

multi db "1",0

section .bss

result: resb 1

section .text

global \_start

\_start:

mov eax, write

mov ebx, stdout

mov ecx, prompt1

mov edx, prompt1len

int 80h

;3

mov al, "3"

sub al, '0'

mov bl, [multi]

sub bl, '0'

mul bl

add al, '0'

mov [result], al

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

mov eax, [multi]

sub eax, '0'

mov ebx, '1'

add eax, ebx

mov [multi], eax

int 80h

;6

mov al, "3"

sub al, '0'

mov bl, [multi]

sub bl, '0'

mul bl

add al, '0'

mov [result], al

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, write

mov ebx, stdout

mov ecx, comma

mov edx, 1

int 80h

mov eax, [multi]

sub eax, '0'

mov ebx, '1'

add eax, ebx

mov [multi], eax

int 80h

;9

mov al, "3"

sub al, '0'

mov bl, [multi]

sub bl, '0'

mul bl

add al, '0'

mov [result], al

int 80h

mov eax, write

mov ebx, stdout

mov ecx, result

mov edx, 1

int 80h

mov eax, [multi]

sub eax, '0'

mov ebx, '1'

add eax, ebx

mov [multi], eax

int 80h

mov eax, write

mov ebx, stdout

mov ecx, newline

mov edx, 1

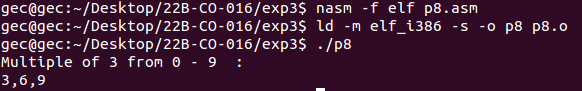
int 80h

mov eax, exit

mov ebx, close

int 80h

**Output:**

****