**Code:**

section .data

input db 'Enter number:'

ilen equ $-input

output db 'Largest number is:'

olen equ $-output

section .bss

num1 resb 4

num2 resb 4

large resb 4

section .text

global \_start:

\_start:

;Input number 1

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num1

mov edx,4

int 0x80

;Input number 2

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num2

mov edx,4

int 0x80

;Print output

mov eax,4

mov ebx,1

mov ecx,output

mov edx,olen

int 0x80

;Find larger number

mov eax,[num1]

mov ebx,[num2]

CMP eax,ebx

JG L1

JMP L2

L1:

mov [large],eax

JMP Print

L2:

mov [large],ebx

JMP Print

;Print larger number

Print:

mov eax,4

mov ebx,1

mov ecx,large

mov edx,4

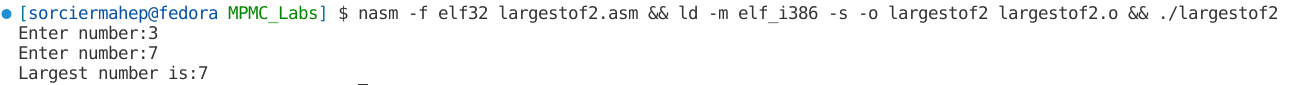
int 0x80

mov eax,1

xor ebx,ebx

int 0x80

**Output:**

****

**Code:**

section .data

input db 'Enter number:'

ilen equ $-input

output db 'Largest number is:'

olen equ $-output

section .bss

num1 resb 4

num2 resb 4

num3 resb 4

large resb 4

section .text

global \_start:

\_start:

;Input number 1

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num1

mov edx,4

int 0x80

;Input number 2

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num2

mov edx,4

int 0x80

;Input number 3

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num3

mov edx,4

int 0x80

;Print output

mov eax,4

mov ebx,1

mov ecx,output

mov edx,olen

int 0x80

;Find larger number

mov eax,[num1]

mov ebx,[num2]

mov ecx,[num3]

CMP eax,ebx

JG L1

JLE L2

L1:

CMP eax,ecx

JG L3

JMP L4

L2:

CMP ebx,ecx

JG L5

JMP L4

L3:

mov [large],eax

JMP Print

L4:

mov [large],ecx

JMP Print

L5:

mov [large],ebx

JMP Print

;Print larger number

Print:

mov eax,4

mov ebx,1

mov ecx,large

mov edx,4

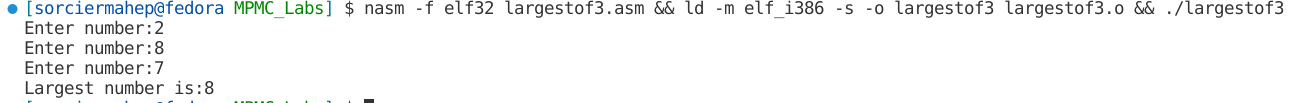
int 0x80

mov eax,1

xor ebx,ebx

int 0x80

**Output:**



**Code:**

section .data

input db 'Enter number:'

ilen equ $-input

output db 'Smallest number is:'

olen equ $-output

section .bss

num1 resb 4

num2 resb 4

small resb 4

section .text

global \_start:

\_start:

;Input number 1

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num1

mov edx,4

int 0x80

;Input number 2

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num2

mov edx,4

int 0x80

;Print output

mov eax,4

mov ebx,1

mov ecx,output

mov edx,olen

int 0x80

;Find smaller number

mov eax,[num1]

mov ebx,[num2]

CMP eax,ebx

JG L1

JMP L2

L1:

mov [small],ebx

JMP Print

L2:

mov [small],eax

JMP Print

;Print smaller number

Print:

mov eax,4

mov ebx,1

mov ecx,small

mov edx,4

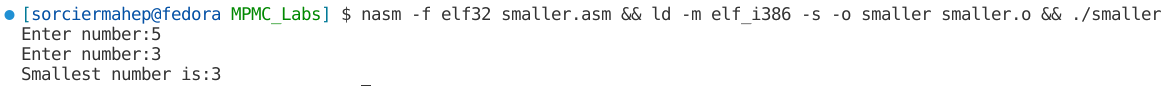
int 0x80

mov eax,1

xor ebx,ebx

int 0x80

**Output:**

****

**Code:**

section .data

input db 'Enter number:'

ilen equ $-input

less db 'Num less than 5',10

lesslen equ $-less

equal db 'Num equal to 5',10

equallen equ $-equal

more db 'Num more than 5',10

morelen equ $-more

output db 'Largest number is:'

olen equ $-output

section .bss

num resb 4

text resw 20

section .text

global \_start:

\_start:

;Input number 1

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num

mov edx,4

int 0x80

;Compare with 5

mov al,[num]

sub al,'0'

mov bl,'5'

sub bl,'0'

CMP al,bl

JE L1

JG L2

JMP L3

L1:

mov eax,4

mov ebx,1

mov ecx,equal

mov edx,equallen

int 0x80

JMP L4

L2:

mov eax,4

mov ebx,1

mov ecx,more

mov edx,morelen

int 0x80

JMP L4

L3:

mov eax,4

mov ebx,1

mov ecx,less

mov edx,lesslen

int 0x80

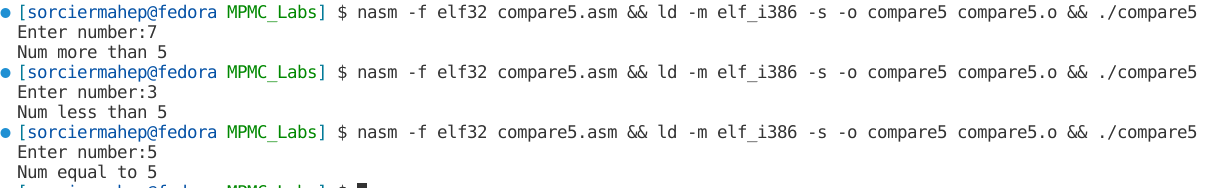
L4:

mov eax,1

xor ebx,ebx

int 0x80

**Output:**

****

**Code:**

section .data

input db 'Enter number:'

ilen equ $-input

even db 'Number is even',10

evenlen equ $-even

odd db 'Number is odd',10

oddlen equ $-odd

section .bss

num resb 4

text resw 20

section .text

global \_start:

\_start:

;Input number 1

mov eax,4

mov ebx,1

mov ecx,input

mov edx,ilen

int 0x80

mov eax,3

mov ebx,2

mov ecx,num

mov edx,4

int 0x80

;Compare with 5

mov al,[num]

sub al,'0'

mov bl,'2'

sub bl,'0'

div bl

CMP al,bl

JE L1

JMP L2

L1:

mov eax,4

mov ebx,1

mov ecx,odd

mov edx,oddlen

int 0x80

JMP L3

L2:

mov eax,4

mov ebx,1

mov ecx,even

mov edx,evenlen

int 0x80

L3:

mov eax,1

xor ebx,ebx

int 0x80

**Output:**

