;Programmer title : ALP using nasm on linux to read and display array elements

%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

section .data

arrsize equ 7

msg1 db "Enter 7 characters : ",10,13

msglen1 equ $-msg1

msg2 db "Entered characters are : ",10,13

msglen2 equ $-msg2

newline db '',10,13

space db ' '

section .bss

arr resb 7

extr resb 1

section .text

global \_start:

\_start:

write msg1,msglen1

mov edi,arr

mov ecx,arrsize

input :

push ecx

read edi,1

read extr,1

inc edi

pop ecx

loop input

write newline,1

write msg2,msglen2

mov esi,arr

mov ecx,arrsize

output:

mov [buff],ecx ; push ecx

write esi,1

write space,1

inc esi

mov ecx,[buff] ; pop ecx

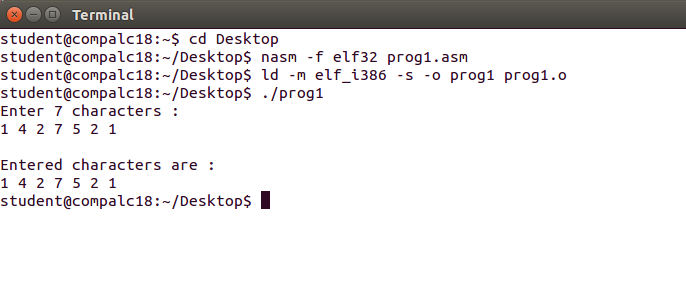
loop output

write newline,1

mov eax,1

mov ebx,0

int 80h



;Programmer title : ALP using nasm on linux to transfer block of data from one location to another

;Programmmer Name : Kevin Francis Fernandes

%macro write 2

mov eax,4

mov ebx,1

mov ecx,%1

mov edx,%2

int 80h

%endmacro

section .data

msg1 db 10,13,"Before transfer the array contains the elements : "

msglen1 equ $-msg1

msg2 db 10,13,"After transfer the array contains the elements : "

msglen2 equ $-msg2

array db 11h,59h,33h,22h,44h

darr db 00h,00h,00h,00h,00h

newline db '',10,13

section .bss

counter resb 1

result resb 4

section .text

global \_start

\_start :

write msg1,msglen1

mov byte[counter],05

mov esi,array

next: mov al,[esi]

push esi

call disp

pop esi

inc esi

dec byte[counter]

jnz next

mov byte[counter],05

mov esi ,array

mov edi ,darr

next2: mov al ,[esi]

mov [edi],al

inc esi

inc edi

dec byte[counter]

jnz next2

write msg2,msglen2

mov byte[counter],05

mov edi,darr

next3:mov al,[edi]

push edi

call disp

pop edi

inc edi

dec byte[counter]

jnz next3

mov eax,1

mov ebx,0

int 80h

;Procedure to display 2 digit numbers

disp : mov bl,al

mov edi,result

mov cx,02

again : rol bl,04

mov al,bl

and al,0fh

cmp al,09

jg down

add al,30h

jmp skip1

down : add al,37h

skip1 : mov [edi],al

inc edi

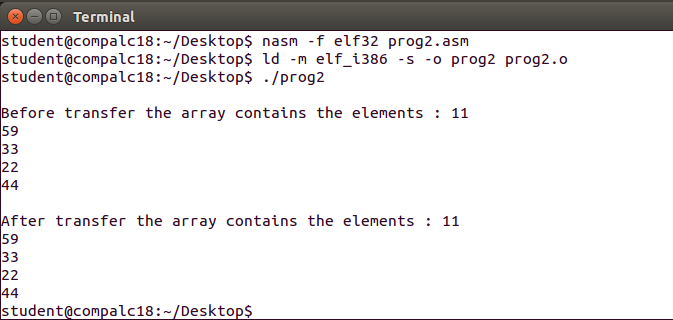
dec cx

jnz again

write result,2

write newline,2

ret



;Programmer title : ALP using nasm on linux to implement a bubble sort algorithm

;Programmmer Name : Kevin Francis Fernandes

%macro write 2

mov eax,4

mov ebx,1

mov ecx,%1

mov edx,%2

int 80h

%endmacro

section .data

msg1 db 10,"The array contains the elements",10,13

msglen1 equ $-msg1

msg2 db 10,13,"The Sorted Array is : ",10,13

msglen2 equ $-msg2

array db 05h,01h,03h,02h,08h

rem db 00h,00h,00h,00h,00h

newline db 13,10

section .bss

counter resb 1

counter2 resb 1

result resb 4

section .text

global \_start

\_start:

write msg1,msglen1

mov byte[counter],05

mov esi,array

next: mov al,[esi]

push esi

call disp

pop esi

inc esi

dec byte[counter]

jnz next

mov byte[counter],05

mov esi,array

mov edi,rem

next2:

mov al,[esi]

mov [edi],al

inc esi

inc edi

dec byte[counter]

jnz next2

mov byte[counter],04

next3:

mov esi,rem

mov edi,rem

mov bl,[esi]

inc esi

mov byte[counter2],04

next4:

mov al,bl

mov bl,[esi]

cmp al,bl

jg great

inc esi

inc edi

test:

dec byte[counter2]

jnz next4

dec byte[counter]

jnz next3

write msg2,msglen2

mov byte[counter],05

mov esi,rem

next5 : mov al,[esi]

push esi

call disp

pop esi

inc esi

dec byte[counter]

jnz next5

mov eax,1

mov ebx,0

int 80h

disp:

mov bl,al

mov edi,result

mov cx,02

again:

rol bl,4

mov al,bl

and al,0fh

cmp al,09

jg down

add al,30h

jmp skip1

down: add al,37h

skip1: mov[edi],al

inc edi

dec cx

jnz again

write result,2

write newline,2

ret

great:

mov [edi],bl

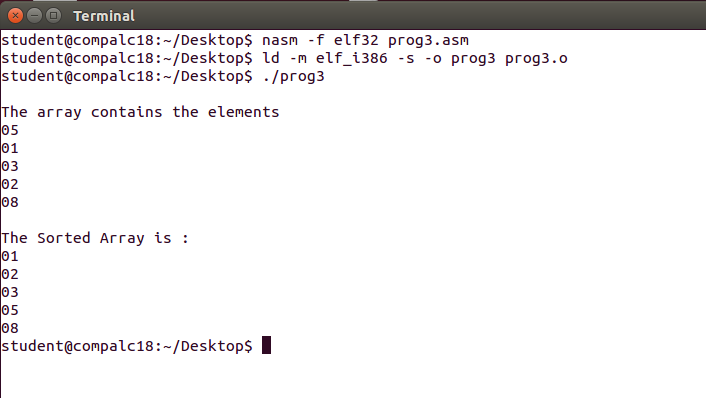
inc edi

mov[edi],al

inc esi

mov bl,al

jmp test



;Programmer title : ALP using nasm on linux to implement an insertion sort algorithm

;Programmmer Name : Kevin Francis Fernandes

%macro display 2

mov edx,%2

mov ecx,%1

mov ebx,1

mov eax,4

int 80h

%endmacro

section .data

msg1 db 10,"The elements in array before sorting",10

len1 equ $-msg1

msg2 db 10,"The elements in array after sorting",10

len2 equ $-msg2

array db 50h,15h,45h,20h,10h

newline db 10

section .bss

result resb 2

count resb 1

cnt resb 1

counter resb 1

section .text

global \_start

\_start: xor eax,eax

mov al,00h

mov [count],al

display msg1,len1

call convert

mov esi,array

mov edi,array

xor eax,eax

mov al,01h

mov [counter],al

up1: inc esi

xor ecx,ecx

mov cl,[counter]

xor eax,eax

mov al,[esi]

up: xor ebx,ebx

mov bl,[edi]

cmp bl,al

ja next

jmp down

next:

inc edi

mov [edi],bl

dec edi

mov [edi],al

dec edi

loop up

down:

inc byte[counter]

mov edi,array

xor ecx,ecx

mov cl,[counter]

sub cl,01h

add edi,ecx

inc byte[count]

cmp byte[count],4 ;Iterate till count == 4

jne up1

display msg2,len2

call convert

mov eax,1

mov ebx,0

int 80h

convert:

xor eax,eax

mov al,05h

mov [cnt],al

mov esi,array

above1:

mov edi,result

xor ecx,ecx

mov cl,02h

mov bl,[esi]

above:

rol bl,04

mov al,bl

and al,0fh

cmp al,09h

jbe below

add al,07h

below:

add al,30h

mov [edi],al

inc edi

loop above

display result,2

display newline,1

inc esi

dec byte[cnt]

cmp byte[cnt],0

jne above1

ret



;Programmer title : ALP using nasm on linux to implement an selection sort algorithm

;Programmmer Name : Kevin Francis Fernandes

%macro display 2

mov edx,%2

mov ecx,%1

mov ebx,1

mov eax,4

int 80h

%endmacro

section .data

msg1 db 10,"The elements in array before sorting"

len1 equ $-msg1

msg2 db 10,13,"The elements in array after sorting"

len2 equ $-msg2

array db 45h,25h,35h,15h,10h

newline db 10

section .bss

result resb 2

count resb 1

cnt resb 1

section .text

global \_start

\_start: xor eax,eax

mov al,00h

mov [count],al

display msg1,len1

call convert

call selection\_sort

display msg2,len2

call convert

mov eax,1

mov ebx,0

int 80h

selection\_sort:

mov esi,array

up1: xor ecx,ecx

mov cl,04h

sub cl,[count]

mov edi,esi

xor eax,eax

mov al,[esi]

up:

inc edi

xor ebx,ebx

mov bl,[edi]

cmp al,bl ;compare al and bl

ja next ;If al>bl go to lable 'next'

jmp next1 ;jump to lable next1

next: mov al,[edi]

next1: loop up

mov edi,array

dec edi

till:

inc edi

cmp [edi],al

jne till

xor ebx,ebx

mov bl,[esi]

mov [esi],al

mov [edi],bl

inc esi

inc byte[count]

cmp byte[count],4

jne up1

ret

convert:

xor eax,eax

mov al,05h

mov [cnt],al

mov esi,array

above1: mov edi,result

xor ecx,ecx

mov cl,02h

mov bl,[esi]

above: rol bl,04

mov al,bl

and al,0fh

cmp al,09h

jbe below

add al,07h

below:

add al,30h

mov [edi],al

inc edi

loop above

display newline,1

display result,2

inc esi

dec byte[cnt]

cmp byte[cnt],0

jne above1

ret

