Assembly Programs:

1. Write an alp to implement linear search

section .data

msg1 db 'Enter number of elements : '

msg1len equ $-msg1

msg2 db 'Enter elements in array : '

msg2len equ $-msg2

msg3 db 'Enter number to be searched: '

msg3len equ $-msg3

msg4 db 'Number found at index: '

msg4len equ $-msg4

msg5 db 'Number not found'

msg5len equ $-msg5

newline db '',10

n1 equ $-newline

%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, 2

mov ecx, %1

mov edx, %2

int 80h

%endmacro

input:

mov byte[i], 0

mov esi, arr

l\_input:

read element, 2

mov ebx, [element]

sub ebx, '0'

mov [esi], ebx

inc esi

inc byte[i]

mov al, [i]

mov bl, [num]

sub bl, '0'

CMP al, bl

JL l\_input

mov byte[i], 0

mov esi, arr

ret

linear\_search:

mov ecx, [num]

sub ecx, '0'

mov esi, 0

label:

movzx edi, byte[arr+esi]

mov [temp], edi

mov al, [temp]

mov bl, [searche]

sub bl, '0'

cmp al, bl

je l1

jmp end

l1:

write msg4, msg4len

add esi, '0'

mov [index], esi

write index, 2

write newline, n1

mov eax, 1

mov ebx, 0

int 80h

end:

inc esi

loop label

ret

section .bss

arr resb 10

searche resb 10

i resb 1

index resb 2

element resb 2

num resb 10

temp resb 5

section .text

global \_start

\_start:

write msg1, msg1len

read num, 10

write msg2, msg2len

call input

write msg3, msg3len

read searche, 10

call linear\_search

write msg5, msg5len

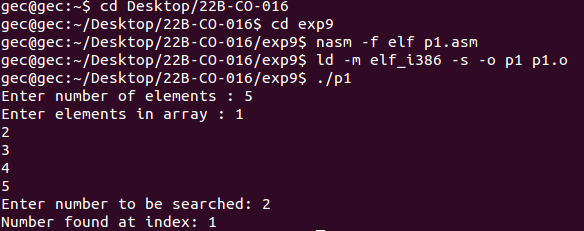
write newline, n1

mov eax, 1

mov ebx, 0

int 80h

**Output:**

****

2. Write an alp to implement binary search

section .data

nl db "", 10

nllen equ $-nl

asksize db 'Enter the number of elements: '

asksizelen equ $-asksize

ask db 'Enter the elements'

asklen equ $-ask

aski db 'Enter the element to be searched: '

askilen equ $-aski

show db 'Array: '

showlen equ $-show

space db ' '

spacelen equ $-space

f db 'Element found at index: '

flen equ $-f

nf db 'Element not found'

nflen equ $-nf

array times 10 dw 0

showl db 'LB: '

showllen equ $-showl

showu db 'UB: '

showulen equ $-showu

showm db 'MID: '

showmlen equ $-showm

; WRITE MACRO

%macro write 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

; READ MACRO

%macro read 2

mov eax, 3

mov ebx, 2

mov ecx, %1

mov edx, %2

int 80h

%endmacro

; NEWLINE MACRO

%macro endl 0

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

%endmacro

; DECLARE VARIABLES

section .bss

num resb 1

counter resb 1

temp resb 1

element resb 1

lb resb 1

ub resb 1

mid resb 1

section .text

global \_start

\_start:

write asksize, asksizelen

read num, 2

write ask, asklen

endl

mov esi, array

mov byte[counter], 0

input:

read temp, 2

mov al, [temp]

mov [esi], al

inc esi

inc byte[counter]

mov al, [num]

sub al, '0'

mov bl, [counter]

cmp al, bl

je ask\_element

jmp input

ask\_element:

write aski, askilen

read element, 2

mov byte[counter], 0

mov esi, array

display:

mov al, [counter]

mov bl, [num]

sub bl, '0'

cmp al, bl

je cont

write esi, 1

write space, spacelen

inc esi

inc byte[counter]

jmp display

cont:

endl

mov al, 0

mov [lb], al

mov bl, [num]

sub bl, '0'

sub bl, 1

mov [ub], bl

check:

mov al, [lb]

mov bl, [ub]

cmp al, bl

jg failure

add al, bl

mov bl, 2

div bl

mov [mid], al

; code to show low, mid, high

write showl, showllen

mov al, [lb]

add al, '0'

mov [temp], al

write temp, 1

endl

write showu, showulen

mov al, [ub]

add al, '0'

mov [temp], al

write temp, 1

endl

write showm, showmlen

mov al, [mid]

add al, '0'

mov [temp], al

write temp, 1

endl

endl

;code to show low, mid, high ends

mov esi, array

mov byte[counter], 0

movesi:

mov al, [counter]

mov bl, [mid]

cmp al, bl

je midcheck

inc esi

inc byte[counter]

jmp movesi

midcheck:

mov al, [esi]

sub al, '0'

mov bl, [element]

sub bl, '0'

cmp al, bl

je found

jl greater

jg lesser

greater:

mov al, [mid]

add al, 1

mov [lb], al

jmp check

lesser:

mov al, [mid]

sub al, 1

mov [ub], al

jmp check

found:

write f, flen

mov al, [mid]

add al, '0'

mov [temp], al

write temp, 1

endl

jmp exit

; EXIT CALL

failure:

write nf, nflen

endl

exit:

mov eax, 1

mov ebx, 0

int 80h

**Output:**

