%macro write 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

%macro read 2

mov eax, 3

mov ebx, 0

mov ecx, %1

mov edx, %2

int 0x80

%endmacro

section .data

msg1 db "Enter number of elements : "

len1 equ $ - msg1

msg2 db 0xA, 0xD, "Enter the array: ", 0xA, 0xD

len2 equ $ - msg2

msg3 db 0xA, 0xD, "Enter the value to search: "

len3 equ $ - msg3

msg4 db 0xA, 0xD, "Element found at index :"

len4 equ $ - msg4

msg5 db 0xA, 0xD, "Element not found."

len5 equ $ - msg5

newline db 0xA, 0xD

section .bss

array resb 100

arrlen resb 1

num\_ascii resb 2

num\_value resb 1

dispbuffer resq 1

section .text

global \_start

\_start:

write msg1, len1 ; accept no. of elements

read num\_ascii, 2

call ASCII\_TO\_HEX

mov eax, [num\_value]

mov [arrlen], eax

write msg2, len2 ; accept elements of array

xor ecx, ecx

mov cl, [arrlen]

mov edi, array

accept\_elms:

pusha

read num\_ascii, 2

call ASCII\_TO\_HEX

popa

mov al, [num\_value]

mov [edi], al

; loop until last element

inc edi

loop accept\_elms

; accept element to search for

write msg3, len3

read num\_ascii, 2

call ASCII\_TO\_HEX

mov al, [num\_value] ; search value

xor ecx, ecx

mov cl, [arrlen]

mov esi, array

search\_elm:

mov ebx, [esi]

cmp al, bl

je match\_found

; loop till end of array

inc esi

loop search\_elm

match\_not\_found:

write msg5, len5

jmp exit

match\_found:

pusha

write msg4, len4

popa

mov ebx, esi

sub ebx, array

call display

exit:

mov eax, 1

int 0x80

ASCII\_TO\_HEX:

mov esi, num\_ascii

mov edi, num\_value

mov ecx, 0x02

xor eax, eax

xor ebx, ebx

up:

rol bl, 0x04

mov al, [esi] ; move ascii char to al

cmp al, 0x39 ; is it a numerical char?

jbe skipc

sub al, 0x07 ; if not, convert letters A-F

skipc:

sub al, 0x30 ; convert char to numerical value

add bl, al ; add it to bl

mov [edi], bl ; store bl in num\_value location

inc esi

loop up

ret

display:

mov ecx, 8

mov edi, dispbuffer

dup:

rol ebx, 04

mov al, bl

and al, 0x0f

cmp al, 0x09

jbe next

add al, 0x07

next:

add al, 0x30

mov [edi], al

inc edi

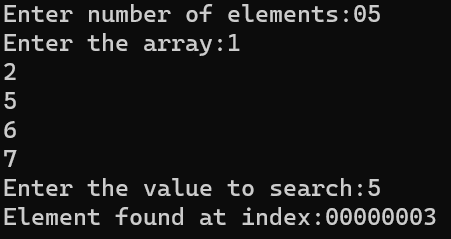
loop dup

write dispbuffer, 8

ret

OUTPUT:

successful



unsuccessful

