**МІНІСТЕРСТВО ОСВІТИ І НАУКИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАІНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

**КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І СПЕЦІАЛІЗОВАНИХ КОМП`ЮТЕНИХ СИСТЕМ**

**ЛАБОРАТОРНАЯ РОБОТА №7**

з дисципліни «**Системне програмування**»

Тема: «**Вивчення основ програмування в захищеному режимі, дослідження механізмів захисту програмних об'єктів**»

|  |  |
| --- | --- |
|  | Виконав  студент 2 курсу  гр. КВ-43  Вітовщик Михайло Васильович  Перевірив(ла):  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2016

***Постановка задачі***

***Варіант 21***

1. Перевірка захисту від виходу за межі сегменту даних.

7. Перевірка присутності сегменту.

8. Перевірка захисту від завантаження в сегментний регістр селектора дескриптора сегмента іншого типу.

***Реалізація тестів***

.386p

descriptor struc

limit\_1 dw 0 ; Limit (0 - 15).

base\_1 dw 0 ; Base (0 - 15).

base\_2 db 0 ; Base (16 - 23).

attr db 0 ; Attributes.

bt6 db 0 ; Limit\_2 (0 - 3), Attributes (4 - 7).

base\_3 db 0 ; Base (24 - 31).

descriptor ends

;----------------------------------------------------------------------

dt\_pointer struc

limit dw 0

address dd 0

dt\_pointer ends

;----------------------------------------------------------------------

attr\_code\_seg equ 10011010b

attr\_data\_seg equ 10010010b

attr\_ro\_data\_seg equ 10010000b

attr\_data\_dpl equ 10110010b

S\_sw0 equ 10010110b ; Writable stack segment

;----------------------------------------------------------------------

\_ST segment use16

db 1000 dup (0)

Top\_stp equ $

\_ST ends

;----------------------------------------------------------------------

gdt segment para public 'data' use16

assume ds: gdt

selector0 descriptor <>

descriptor\_gdt descriptor <> ; global descriptors

descriptor\_ds descriptor <> ; \_data

descriptor\_es descriptor <> ; 0b8000h

descr\_ss descriptor <>

descriptor\_undefined descriptor <>

descriptor\_cs descriptor <> ;code

descriptor\_int\_code descriptor <> ; int code

descriptor\_absent descriptor <>

descriptor\_fs descriptor <> ; try

descriptor\_gs descriptor <> ; try

descr\_t7 descriptor <>

descr\_t14 descriptor <>

gdt\_size equ $ - selector0 - 1

gdt ends

;----------------------------------------------------------------------

idt segment para public 'data' use16

vector = 0

rept 256

dw vector \* proc\_int\_size ; çì³ùåííÿ ïðîöåäóðè

dw offset descriptor\_int\_code ; ñåëåêòîð äåñêðèïòîðà â GDT

db 0 ; çàâæäè 0

db 10001111b ; áàéò àòðèáóò³â - øëþç ïàñòêè; change dpl=1

dw 0 ; âñ³ ïðîöåäóðè ðîçì³ùóþòüñÿ â ïåðøèõ 64Ê

vector = vector + 1

endm

idt ends

;----------------------------------------------------------------------

\_readonly\_data segment public 'data' use16

assume fs: \_readonly\_data

val dd 0

readonly\_data\_size = 4

\_readonly\_data ends

;----------------------------------------------------------------------

\_dpl1\_data segment public 'data' use16

assume gs:\_dpl1\_data

just\_something dd ?

dpl\_size = $ - \_dpl1\_data

\_dpl1\_data ends

;----------------------------------------------------------------------

\_data segment public 'data' use16

assume ds: \_data

data\_start label dt\_pointer

gdt\_pointer dt\_pointer <>

idt\_pointer dt\_pointer <>

Top\_real\_mode dd 0

message db "Hello from protected mode!"

message\_len = $ - message

data\_size = $ - data\_start

\_data ends

;----------------------------------------------------------------------

\_interrupt\_code segment para public 'code' use16

assume cs: \_interrupt\_code

vector = 0

int\_begin:

pusha

push bx

push 2480

call show

mov ax, vector

jmp common\_int

int\_end:

proc\_int\_size = int\_end - int\_begin

vector = 1

rept 255

pusha

push bx

push 2480

call show

mov ax, vector

jmp common\_int

vector = vector + 1

endm

int\_msg db 'INTERRUPT N'

common\_int:

cld

mov cl, 10

div cl ; ìîëîäøà öèôðà â ah

or ah, 30h ; ôîðìóâàííÿ òåêñòîâîãî êîäó öèôðè

mov bh, ah

xor ah, ah

div cl ; ñåðåäíÿ öèôðà â ah, ñòàðøà - â al

or ax, 3030h ; ôîðìóâàííÿ òåêñòîâîãî êîäó öèôð

mov dx, ax

; âèâåäåííÿ ïîâ³äîìëåííÿ íà åêðàí

push offset descriptor\_es ; ñåëåêòîð äåñêðèïòîðà ³ç 1-ãî åòàïó

pop es

mov si, offset int\_msg

mov cx, 11

mov di, 2620 ;ïî÷àòêîâà àäðåñà äëÿ âèâåäåííÿ

mov ah, 07h ;àòðèáóò ñèìâîë³â

outstr:

mov al, cs:[si]

stosw

inc si

loop outstr

mov al, ' '

stosw

mov al, dl

stosw

mov al, dh

stosw

mov al, bh

stosw

pushf

push offset descriptor\_cs

push offset dos\_return

iret

show proc near

push bp

mov bp,sp

pusha

push es

mov ax,offset descriptor\_es

mov es,ax

std

ls20:

mov di,[bp+4]

mov bx,[bp+6]

mov cx,4

mov ah,0ah

ls40:

mov al,bl

and al,00001111b

cmp al,10

jl ls100

add al,7

ls100:

add al,30h

stosw

shr bx,4

loop ls40

pop es

popa

pop bp

ret 4

show endp

common\_int\_end:

int\_code\_size = common\_int\_end - int\_begin

\_interrupt\_code ends

;----------------------------------------------------------------------

\_code segment byte public 'code' use16

assume cs: \_code

main:

mov ax, \_data

mov ds,ax

mov ax,3h

int 10h

mov eax, cr0

test al, 1

jnz already\_protected

; Create descriptors.

mov ax, gdt

mov ds, ax

assume ds: gdt

; GDT descriptor.

mov word ptr [descriptor\_gdt.limit\_1], gdt\_size

xor eax, eax

mov ax, gdt

shl eax, 4

mov dword ptr [descriptor\_gdt.base\_1], eax

mov byte ptr [descriptor\_gdt.attr], attr\_data\_seg

; Code descriptor.

mov word ptr [descriptor\_cs.limit\_1], code\_size

xor eax, eax

mov ax, \_code

shl eax, 4

mov dword ptr [descriptor\_cs.base\_1], eax

mov byte ptr [descriptor\_cs.attr], attr\_code\_seg

; DS descriptor.

mov word ptr [descriptor\_ds.limit\_1], data\_size

xor eax, eax

mov ax, \_data

shl eax, 4

mov dword ptr [descriptor\_ds.base\_1], eax

mov byte ptr [descriptor\_ds.attr], attr\_data\_seg

; \_ST segment descr

mov descr\_ss.limit\_1, 10

xor eax, eax

mov ax, \_ST

shl eax, 4

mov dword ptr descr\_ss.base\_1, eax

mov descr\_ss.attr, S\_sw0

; absent descriptor.

mov word ptr [descriptor\_absent.limit\_1], data\_size

xor eax, eax

mov ax, \_data

shl eax, 4

mov dword ptr [descriptor\_absent.base\_1], eax

mov byte ptr [descriptor\_absent.attr], attr\_data\_seg

and byte ptr [descriptor\_absent.attr], 01111111b

; GS descriptor.

mov word ptr [descriptor\_gs.limit\_1], dpl\_size

xor eax, eax

mov ax, \_dpl1\_data

shl eax, 4

mov dword ptr [descriptor\_gs.base\_1], eax

mov byte ptr [descriptor\_gs.attr], attr\_data\_dpl

; ES descriptor.

mov word ptr [descriptor\_es.limit\_1], 0ffffh

mov dword ptr [descriptor\_es.base\_1], 0b8000h

mov byte ptr [descriptor\_es.attr], attr\_data\_seg

; Interrupt descriptor

mov word ptr [descriptor\_int\_code.limit\_1], int\_code\_size

xor eax, eax

mov ax, \_interrupt\_code

shl eax, 4

mov dword ptr [descriptor\_int\_code.base\_1], eax

mov byte ptr [descriptor\_int\_code.attr], attr\_code\_seg

; FS descriptor

mov word ptr [descriptor\_fs.limit\_1], readonly\_data\_size

xor eax, eax

mov ax, \_readonly\_data

shl eax, 4

mov dword ptr [descriptor\_fs.base\_1], eax

mov byte ptr [descriptor\_fs.attr], attr\_ro\_data\_seg

and descriptor\_fs.attr, 11111000b

;----------------------------------

mov descr\_t7.attr, 00011010b

mov descr\_t14.limit\_1, data\_size

xor eax, eax

mov ax, \_data

shr eax, 4

mov dword ptr descr\_t14.base\_1, eax

mov descr\_t14. attr, attr\_ro\_data\_seg

;------------------------------------

; Load GDTR.

mov ax, \_data

mov ds, ax

assume ds: \_data

; Create gdt pointer.

xor eax, eax

mov ax, gdt

shl eax, 4

mov dword ptr ds:[gdt\_pointer.address], eax

mov ax, gdt\_size

mov word ptr ds:[gdt\_pointer.limit], ax

lgdt gdt\_pointer

; Create idt pointer.

xor eax, eax

mov ax, idt

shl eax, 4

mov dword ptr [idt\_pointer.address], eax

mov word ptr [idt\_pointer.limit], 8 \* 256

lidt idt\_pointer

cli

; Go to protected mode.

mov word ptr Top\_real\_mode, sp

mov word ptr Top\_real\_mode+2, ss

mov eax, cr0

or eax, 1

mov cr0, eax

; Long jmp to protected.

db 0eah

dw offset protected

dw offset descriptor\_cs

protected:

; Load other segment registers.

mov ax,offset descr\_ss

mov ss,ax

mov sp,Top\_stp

; ds.

mov ax, offset descriptor\_ds

mov ds, ax

; es.

mov ax, offset descriptor\_es

mov es, ax

; fs.

mov ax, offset descriptor\_fs

mov fs, ax

; gs.

mov ax, offset descriptor\_gs

mov gs, ax

;jmp dos\_return

; Print message.

mov cx, message\_len

mov si, offset message

mov di, 0

mov ah, 07h

message\_printing:

mov al, byte ptr ds:[si]

mov word ptr es:[di], ax

inc si

add di, 2

loop message\_printing

;===================================================

;test 1

mov bx, data\_size+1

mov al, [bx]

;test 7

;mov ax, offset descr\_t7

;mov ds, ax

;test 8

;mov ax, offset descr\_cs

;mov ds, ax

;test 14

;assume ds: gdt

;mov ax, offset decr\_gdt

;mov ds, ax

;mov al, descr\_t14.attr

;or al, 1

;mov descr.attr, al

;mov ax, offset\_t14

;mov ds, ax

;===================================================

dos\_return:

cli

; Return to dos.

assume ds:gdt

mov ax, offset descriptor\_gdt

mov ds, ax

mov descriptor\_cs.limit\_1, 0ffffh

mov descriptor\_ds.limit\_1, 0ffffh

mov descriptor\_es.limit\_1, 0ffffh

mov descriptor\_fs.limit\_1, 0ffffh

mov descriptor\_fs.attr, attr\_data\_seg

mov descr\_ss.limit\_1, 0ffffh

mov descr\_ss.attr, attr\_data\_seg

mov descriptor\_gs.limit\_1, 0ffffh

; Reload segment registers.

mov ax, offset descriptor\_ds

mov ax, ds

mov ax, offset descriptor\_es

mov es, ax

mov ax, offset descriptor\_fs

mov fs, ax

mov ax, offset descriptor\_gs

mov gs, ax

mov ax, offset descr\_ss

mov ss, ax

; Restore old interrupt table.

mov ax, offset descriptor\_ds

mov ds, ax

assume ds: \_data

xor eax, eax

mov dword ptr ds:[idt\_pointer.address], eax

mov word ptr ds:[idt\_pointer.limit], 3ffh

lidt ds:[idt\_pointer]

db 0eah

dw offset descriptor\_cs\_reload

dw offset descriptor\_cs

descriptor\_cs\_reload:

mov eax, cr0

and al, 0feh

mov cr0, eax

db 0eah

dw offset real\_mode

dw \_code

real\_mode:

mov ax, \_data

mov ds, ax

assume ds: \_data

lss sp, Top\_real\_mode

mov ax, \_data

mov ds, ax

mov es, ax

mov fs, ax

mov gs, ax

sti

mov ah, 0

int 16h

mov ax, 3

int 10h

already\_protected:

mov ax, 4c00h

int 21h

code\_end:

code\_size = code\_end - main

\_code ends

end main