****

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп’ютерних систем**

**Лабораторна робота №7**

**з дисципліни “Системне програмування”**

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

**Варіант 5**

**Виконав:**

**студент II-го курсу**

**групи КВ-41**

**Горпинич-Радуженко Іван**

**Київ 2016**

**Зміст роботи.**

Робота виконується за два етапи. На першому етапі студентами розробляється та налагоджується програма, яка:

1. Перевіряє поточний режим процесора та підготовлює системні структури, необхідні для роботи в захищеному режимі.
2. Виконує переключення процесора в захищений режим.
3. Виводить діагностичне повідомлення на екран.
4. Забезпечує повернення в режим реальних адрес

Другий етап виконується шляхом доповнення програми, розробленої на першому етапі, системними даними та процедурами для обслуговування переривань (виключень), які виникають в результаті виявлення апаратурою процесора порушень правил захисту програмних об'єктів. За допомогою модифікованої програми проводяться експерименти по активізації захисту згідно завдання.

**Завдання на лабораторну роботу**

2. Перевірка захисту від запису в заборонений для запису сегмент даних.

4. Перевірка захисту від запису в сегмент кодів.

1. Перевірити захист від порушення превілей.
2. Експерементально пересвідчитись, що зміна полів дескриптора в пам'яті без перезаваннтаження селектора не змінює властивостей діючого сегмента.

**Текст програми**

.586p

descr struct

limit\_1 dw 0

base\_1 dw 0

base\_2 db 0

attrib db 0

bt6 db 0

base\_3 db 0

descr ends

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

dt\_p struc

lim dw 0

adr dd 0

dt\_p ends

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

S\_cr0 equ 10011010b ; Readable 0 level code segment

S\_cr1 equ 10111010b ; Readable 1 level code segment

S\_c\_0 equ 10011000b ; UnReadable 0 level code segment

S\_dw0 equ 10010010b ; Writable 0 level data segment

S\_dw1 equ 10110010b ; Writable 1 level data segment

S\_d\_0 equ 10010000b ; UnWritable 0 level data segment

S\_d\_1 equ 10110000b ; UnWritable 1 level data segment

S\_sw0 equ 10010110b ; Writable stack segment

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

\_GDT segment para public 'data' use16

descr\_0 descr <>

descr\_gdt descr <>

descr\_ds descr <>

descr\_es descr <>

descr\_ss descr <>

descr\_cs descr <>

descr\_i\_code descr <>

descr\_t2 descr <>

descr\_t10 descr <>

\_gdt\_size equ $-1

\_GDT ends

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

\_IDT segment para public 'data'

vector = 0

rept 256

dw Vector \* proc\_i\_size

dw offset descr\_i\_code

db 0

db 10001111b

dw 0

vector = vector+1

endm

\_IDT ends

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

\_ST segment use16

db 1000 dup (0)

Top\_stp equ $

\_ST ends

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

\_data segment para public 'data' use16

pgdt dt\_p <>

pidt dt\_p <>

old\_st dd ?

mess db "We are in protected mode!"

messl equ $-mess

t1 db ?

\_data\_size equ $-1

\_data ends

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

\_t2 segment para public 'data' use16

t2\_p:

v1 dw 0

\_t2\_size equ $-t2\_p

\_t2 ends

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

\_code segment para public 'code' use16

assume cs:\_code

\_begin:

mov eax, cr0

test al, 1

jz cont

mov ax, 4C00h

int 21h

cont:

mov ax, 3

int 10h

in al, 92h

or al, 2

out 92h, al

assume ds:\_GDT

mov ax, \_GDT

mov ds, ax

; \_GDT segment descr

mov descr\_gdt.limit\_1, \_gdt\_size

xor eax, eax

mov ax, \_GDT

shl eax, 4

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

mov descr\_gdt.attrib, S\_dw0

;\_data segment descr

mov descr\_ds.limit\_1, \_data\_size

xor eax, eax

mov ax, \_data

shl eax, 4

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

mov descr\_ds.attrib, S\_dw0

; \_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.attrib, S\_sw0

; Video segment descr (es)

mov descr\_es.limit\_1, 0FFFFh

mov dword ptr descr\_es.base\_1, 0B8000h

mov descr\_es.attrib, S\_dw0

; \_code segment descr

mov descr\_cs.limit\_1, \_code\_size

xor eax, eax

mov ax, \_code

shl eax, 4

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

mov descr\_cs.attrib, S\_c\_0

; \_i\_code segment descr

mov descr\_i\_code.limit\_1, \_i\_code\_size

xor eax, eax

mov ax, \_i\_code

shl eax, 4

mov dword ptr descr\_i\_code.base\_1, eax

mov descr\_i\_code.attrib, S\_cr0

; \_t2 segment descr

mov descr\_t2.limit\_1, \_t2\_size

xor eax, eax

mov ax, \_t2

shl eax, 4

mov dword ptr descr\_t2.base\_1, eax

mov descr\_t2.attrib, S\_d\_0

; \_t10 segment descr

mov descr\_t10.limit\_1, \_code\_size

xor eax, eax

mov ax, \_code

shl eax, 4

mov dword ptr descr\_t10.base\_1, eax

mov descr\_t10.attrib, S\_cr1

assume ds:\_data

mov ax, \_data

mov ds, ax

xor eax, eax

mov ax, \_GDT

shl eax, 4

mov pgdt.adr, eax

mov ax, \_gdt\_size

mov pgdt.lim, ax

lgdt pgdt

cli

mov word ptr old\_st, sp

mov word ptr old\_st+2, ss

mov eax, cr0

or al, 1

mov cr0, eax

db 0eah

dw offset protect

dw offset descr\_cs

protect:

mov ax, offset descr\_ss

mov ss, ax

mov sp, Top\_stp

mov ax, offset descr\_ds

mov ds, ax

mov ax, offset descr\_es

mov es, ax

xor eax, eax

mov ax, \_IDT

shl eax, 4

mov pidt.adr, eax

mov pidt.lim, 8\*256

lidt pidt

mov cx, messl

mov si, offset mess

mov di, 1000

mov ah, 07h

outmes:

mov al, [si]

mov es:[di], ax

inc si

add di, 2

loop outmes

;--- Tests ---

; Test 2

; mov ax, offset descr\_t2

; mov ds, ax

; mov bx, 0

; mov byte ptr [bx], al

; Test 4

; mov cs:[0], al

; Test 10

; db 0eah

; dw offset \_begin

; dw offset descr\_t10

; Test 15

; assume ds:\_GDT

; mov ax, offset descr\_t2

; mov gs, ax

; mov ax, offset descr\_GDT

; mov ds, ax

; mov descr\_t2.attrib, S\_dw0

; mov byte ptr gs:v1, 0

;--- End of Tests ---

return\_dos:

assume ds:\_GDT

cli

mov ax, offset descr\_gdt

mov ds, ax

mov descr\_cs.limit\_1, 0FFFFh

mov descr\_ds.limit\_1, 0FFFFh

mov descr\_ss.limit\_1, 0FFFFh

mov descr\_ss.attrib, S\_dw0

assume ds:\_data

mov ax, offset descr\_ds

mov ds, ax

mov gs, ax

mov fs, ax

mov ax, offset descr\_es

mov es, ax

mov ax, offset descr\_ss

mov ss, ax

db 0EAh

dw offset jumpt

dw offset descr\_cs

jumpt:

xor eax, eax

mov pidt.adr, eax

mov pidt.lim, 3FFh

lidt pidt

mov eax, cr0

and al, 0FEh

mov cr0, eax

db 0EAh

dw offset r\_mode

dw \_code

r\_mode:

assume ds:\_data

mov ax, \_data

mov ds, ax

lss sp, old\_st

sti

mov ah, 01h

int 21h

mov ax, 3

int 10h

mov ax, 4C00h

int 21h

\_code\_size equ $-1

\_code ends

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

\_i\_code segment para public 'code' use16

assume cs:\_i\_code

vector = 0

i\_beg:

rept 255

pusha

mov ax, vector

jmp common\_int

vector = vector + 1

endm

i\_end:

proc\_i\_size = 7

mes\_int db 'INTERUPT N'

common\_int:

mov cl, 10

div cl

or ah, 30h

mov bh, ah

xor ah, ah

div cl

or ax, 3030h

mov dx, ax

push offset descr\_es

pop es

mov si, offset Mes\_int

mov cx, 10

mov di, 2620

mov ah, 07h

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 descr\_cs

push offset return\_dos

iret

common\_end:

\_i\_code\_size = $-1

\_i\_code ends

end \_begin