**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський Політехнічний Інститут»**

**Факультет Прикладної Математики**

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

**Лабораторна робота №7**

**З дисципліни: «Системне програмування»**

**Варіант 16**

**Виконав:**

**Студент групи КВ-41**

**Мелащенко O. C.**

**Київ 2016**

**Завдання:**

1. Перевірка захисту від запису в заборонений для запису сегмент даних.
2. Перевірка захисту від запису в сегмент кодів.
3. Перевірка захисту від завантаження в сегментний регістр селектора дескриптора сегмента іншого типу.
4. Перевірити запис 1 в біт **A**

**Текст программи:**

.386p

descr struc

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 code segment

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

S\_dw0 equ 10010010b ; Writable data segment

S\_d\_0 equ 10010000b ; UnWritable 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\_ds\_exp descr <> ;2

descr\_i\_code descr <>

gdt\_size equ $-descr\_0-1

\_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 1000;$

\_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

e\_mod db "Error while checking CR0!"

e\_modl equ $-e\_mod

data\_size equ $-pgdt-1

\_data 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

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

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

mov descr\_ds\_exp.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\_d\_0

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

mov descr\_es.limit\_1, 0FFFFh

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

mov descr\_es.attrib, S\_dw0

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

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

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 esp,1000;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

mov ax,offset descr\_ds\_exp

mov ds,ax

mov bx, 0

mov [bx], ax

mov cs:[0], ax

mov ax,offset descr\_cs

mov ds,ax

assume ds:\_GDT

mov ax, \_GDT

mov ds, ax

mov al,descr\_ds\_exp.attrib

or al, 1

mov descr\_ds\_exp.attrib,al

mov ax,offset descr\_ds\_exp

mov ds,ax

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 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 $-\_begin-1

\_code ends

i\_code segment para public 'code' use16

assume cs:i\_code

vector = 0

i\_beg:

pusha

mov ax, vector

jmp common\_int

i\_end:

proc\_i\_size = i\_end - i\_beg

vector = 1

rept 255

pusha

mov ax, vector

jmp common\_int

vector = vector + 1

endm

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 = common\_end - i\_beg

i\_code ends

end \_begin