КАФЕДРА **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 2 |

**Название:**

Перевод компьютера в защищённый режим

**Дисциплина:** Операционные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Н.А. Павлов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н. Ю. Рязанова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Код программы**

.386p

disable\_interrupts macro

cli

in al, 70h

or al, 80h

out 70h, al

endm

enable\_interrupts macro

in al, 70h

and al, 7fh

out 70h, al

sti

endm

open\_a20 macro

mov al, 0D1h

out 64h, al

mov al, 0DFh

out 60h, al

endm

close\_a20 macro

mov al, 0D1h

out 64h, al

mov al, 0DDh

out 60h, al

endm

far\_jump macro d\_,lbl,selector

db 0EAh

d\_ offset lbl

dw selector

endm

load\_offset macro trp,exception

mov eax, offset exception

mov trp.offs\_l, ax

shr eax, 16

mov trp.offs\_h, ax

endm

cls macro

mov ax,3

int 10h

endm

print\_r macro msg

mov ah, 9

mov edx, offset msg

int 21h

endm

set\_cursor macro offst

mov ah, 2

xor bx, bx

mov dx, offst

int 10h

endm

getchar macro

mov ah, 8

int 21h

endm

print\_string macro msg,msg\_size,row,col, color

local screen

push ebp

xor eax, eax

mov ebp, row \* 160 + col \* 2

add ebp, 0b8000h

mov ecx, msg\_size

mov ah, color

xor esi, esi

screen:

mov al, byte ptr msg[esi]

mov es:[ebp], ax

add ebp, 2

inc esi

loop screen

pop ebp

endm

print\_number macro var, row, col

local cycle, number, print

push ebp

mov eax, var

mov ebp, row \* 160 + col \* 2

mov ecx, 8

add ebp, 0B8000h

cycle:

mov dl, al

and dl, 0Fh

cmp dl, 10

jl number

add dl,'a'-10

jmp print

number:

add dl,'0'

print:

mov es:[ebp], dl

ror eax, 4

sub ebp, 2

loop cycle

pop ebp

endm

segment\_descr struc

limit dw 0

base\_l dw 0

base\_m db 0

attr\_1 db 0

attr\_2 db 0

base\_h db 0

segment\_descr ends

interrupt\_descr struc

offs\_l dw 0

selector dw 0

rsrv db 0

attr db 0

offs\_h dw 0

interrupt\_descr ends

data\_16 segment use16

gdt label byte

gdt\_null descriptor **<>**

gdt\_data16 descriptor **<**data16\_seg\_size**-**1**,**0**,**0**,**92h**,**0**,**0**>**

gdt\_code16 descriptor **<**code16\_seg\_size**-**1**,**0**,**0**,**98h**,**0**,**0**>**

gdt\_data32 descriptor **<**data32\_seg\_size**-**1**,**0**,**0**,**92h**,**40h**,**0**>**

gdt\_code32 descriptor **<**code32\_seg\_size**-**1**,**0**,**0**,**98h**,**40h**,**0**>**

gdt\_stack32 descriptor **<**stack\_size**-**1**,**0**,**0**,**92h**,**40h**,**0**>**

gdt\_screen descriptor **<**4095**,**8000h**,**0Bh**,**92h**>**

gdt\_data\_flat descriptor **<**0FFFFh**,**0**,**10h**,**92h**,**11001111b**,**0**>**

gdt\_size = $-gdt

gdtr dw gdt\_size-1 ; Лимит GDT

dd ? ; Линейный адрес GDT

selector\_data16 equ 8

selector\_code16 equ 16

selector\_data32 equ 24

selector\_code32 equ 32

selector\_stack32 equ 40

selector\_screen equ 48

selector\_data\_flat equ 56

idt label byte

trap\_1 interrupt\_descr 13 dup (<,selector\_code32,,8fh>)

trap\_13 interrupt\_descr <0, selector\_code32,,8fh>

trap\_2 interrupt\_descr 18 dup (<,selector\_code32,,8fh>)

timer interrupt\_descr <,selector\_code32,,8eh>

keyboard interrupt\_descr <,selector\_code32,,8eh>

idt\_size = $-idt

idtr dw idt\_size-1

dd ?

idtr\_r dw 3FFh, 0, 0

msg\_protected\_size = $-msg\_protected

msg\_real db 'Processor works in real mode', '$'

msg\_real\_size = $-msg\_real-1

data16\_seg\_size **=** **$-**data16\_seg

data16\_seg ends

data32\_seg segment use32

data32\_start**:**

msg\_protected db 'Processor works in protected mode'

msg\_timer db 'Timer:'

msg\_timer\_size = $-msg\_timer

msg\_memory db 'Memory:'

msg\_memory\_size = $-msg\_memory

scan\_to\_ascii db 0,1bh,'1','2','3','4','5','6','7','8','9','0','-','=',8

db ' ','q','w','e','r','t','y','u','i','o','p','[',']','$'

db ' ','a','s','d','f','g','h','j','k','l',';','""',0

db '\','z','x','c','v','b','n','m',',','.','/',0,0,0,' ',0,0

db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

screen dd 4 \* 160

tick\_counter dd 0

master db 0

slave db 0

data32\_seg\_size**=$-**data32\_start

data32\_seg ends

code32\_seg segment 'code' use32

assume CS**:**code32\_seg**:**

protected\_entry:

; Загрузка селекторов защищённого режима

mov ax, selector\_data\_flat

mov ds, ax

mov es, ax

mov ax, selector\_stack

mov ebx, \_stack\_size

mov ss, ax

mov esp, ebx

; Разрешение прерываний

enable\_interrupts

print\_string msg\_protected, msg\_protected\_size, 1, 0, 070h

print\_string msg\_memory, msg\_memory\_size, 1, 60, 060h

print\_string msg\_timer, msg\_timer\_size, 22, 60, 050h

call compute\_memory

jmp short $

dummy\_exception proc

iretd

dummy\_exception endp

exception\_13 proc

pop eax

pop eax

iretd

exception\_13 endp

; Обработчик прерывания от таймера

timer\_handler:

push eax

push ecx

push edx

print\_number tick\_counter, 22, 75

; Инкремент и сохранение значения счётчика

inc eax

mov tick\_counter, eax

mov al, 20h

out 20h, al

pop edx

pop ecx

pop eax

iretd

; Обработчик прерывания от клавиатуры

keyboard\_handler:

push eax

push ebx

push es

push ds

in al, 60h

cmp al, 01h

je esc\_pressed

cmp al, 39h

ja skip\_translate

mov bx, selector\_data

mov ds, bx

mov ebx, offset scan\_to\_ascii

; В ASCII

xlatb

mov bx, selector\_data\_flat

mov es, bx

mov ebx, screen

cmp al, 8

je bs\_pressed

mov es:[ebx+0b8000h], al

add dword ptr screen, 2

jmp short skip\_translate

bs\_pressed:

mov al, ' '

sub ebx, 2

mov es:[ebx+0b8000h], al

mov screen, ebx

skip\_translate:

in al, 61h

or al, 80h

out 61h, al

mov al, 20h

out 20h, al

pop ds

pop es

pop ebx

pop eax

iretd

esc\_pressed:

in al, 61h

or al, 80h

out 61h, al

mov al, 20h

out 20h, al

pop ds

pop es

pop ebx

pop eax

disable\_interrupts

db 0eah

dd return\_to\_real

dw selector\_code16

compute\_memory proc

push ds

mov ax, selector\_data\_flat

mov ds, ax

mov ebx, 100001h

; Пропуск первого мегабайта

mov dl, 11101011b

mov ecx, 0FFEFFFFFh

check:

mov dh, ds:[ebx]

mov ds:[ebx], dl

cmp ds:[ebx], dl

jnz end\_of\_memory

mov ds:[ebx], dh

inc ebx

loop check

end\_of\_memory:

pop ds

xor edx, edx

mov eax, ebx

mov ebx, 100000h

div ebx

push ecx

push edx

print\_number eax, 1, 75

pop edx

pop ecx

ret

compute\_memory endp

code\_32\_size = $-gdt

code\_32 ends

code\_16 segment para public 'CODE' use16

assume cs:code\_16, ds:code\_32, ss:\_stack

start\_in\_real:

cls

push code\_32

pop ds

mov ah, 09h

mov edx, offset msg\_real

int 21h

xor eax, eax

mov ax, code\_16

shl eax, 4

mov word ptr gdt\_code16 + 2, ax

shr eax, 16

mov byte ptr gdt\_code16 + 4, al

mov ax, code\_32

shl eax, 4

push eax

push eax

mov word ptr gdt\_code32 + 2, ax

mov word ptr gdt\_stack + 2, ax

mov word ptr gdt\_data + 2, ax

shr eax, 16

mov byte ptr gdt\_code32 + 4, al

mov byte ptr gdt\_stack + 4, al

mov byte ptr gdt\_data + 4, al

pop eax

add eax, offset gdt

; Установка начала GDT

mov dword ptr gdtr + 2, eax

; Установка размера GDT

mov word ptr gdtr, gdt\_size - 1

; Загрузка GDT

lgdt fword ptr gdtr

pop eax

add eax, offset idt

; Установка начала IDT

mov dword ptr idtr + 2, eax

; Установка размера IDT

mov word ptr idtr, idt\_size-1

load\_offset trap\_1, dummy\_exception

load\_offset trap\_13, exception\_13

load\_offset trap\_2, dummy\_exception

load\_offset timer, timer\_handler

load\_offset keyboard, keyboard\_handler

in al, 21h

mov master, al

in al, 0A1h

mov slave, al

; Изменение базового вектора ведущего контроллера прерываний

cli

mov al, 11h

out 20h, al

mov al, 20h

out 21h, al

mov al, 4

out 21h, al

sti

mov al, 1

out 21h, al

; Установка новых масок

mov al, 0FCh

out dx, al

mov dx, 0A1h

mov al, 0FFh

out dx, al

lidt fword ptr idtr

; Открытие линии А20

open\_a20

disable\_interrupts

mov eax, cr0

or al, 1

mov cr0, eax

db 66h

db 0eah

dd offset protected\_entry

dw selector\_code32

return\_to\_real:

close\_a20

; Установка PE = 0

mov eax, cr0

and al, 0FEh

mov cr0, eax

; Загрузка CS

db 0eah

dw $ + 4

dw code\_16

; Регистры

mov ax, code\_32

mov ds, ax

mov es, ax

mov ax, \_stack

mov bx, \_stack\_size

mov ss, ax

mov sp, bx

; Инициализация контроллера прерываний

mov al, 11h

out 20h, al

mov al, 8

out 21h, al

mov al, 4

out 21h, al

mov al, 1

out 21h, al

; Маски

mov al, master

out 21h, al

mov al, slave

out 0A1h, al

; Загрузка регистра IDTR для реального режима

lidt fword ptr idtr\_r

; Разрешение маскируемых и немаскируемых прерываний

enable\_interrupts

set\_cursor 200h

print\_r msg\_real

getchar

cls

mov ah, 4ch

int 21h

code\_16\_size = $-start\_in\_real

code\_16 ends

\_stack segment para stack 'stack'

stack\_start db 100h dup(?)

\_stack\_size = $-stack\_start

\_stack ends

end start\_in\_real

**Демонстрация работы программы**

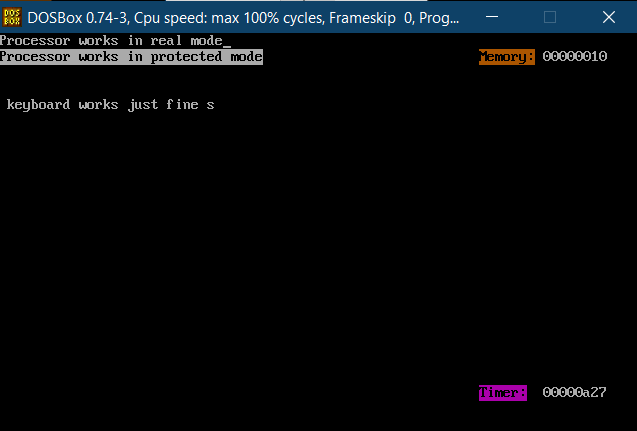
1. Запуск программы. Происходит переход в защищённый режим. Количество выделенной памяти ещё не подсчитано.



1. Через некоторое время количество памяти подсчитывается



1. Проверка обработчика прерывания от клавиатуры



1. Демонстрация работоспособности клавиши **Backspace**



1. Возвращение в реальный режим с помощью клавиши **Esc**



1. Система продолжает корректно работать после возвращения в реальный режим

