**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

---

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **«ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

# О Т Ч Е Т

**по лабораторной работе № 2**

**Название:** Защищенный режим

**Дисциплина:** Операционные системы

Студент ИУ7-53Б _____ П.А.Топорков
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Ю. Рязанова
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

**Задание:**

Написать программу, переводящую компьютер в защищенный режим. Программа начинает работать в реальном режиме. Для перевода в защищенный режим выполняются необходимые действия. В защищенном режиме программа работает на нулевом уровне привилегий.

В защищенном режиме программа должна позволять определить объем доступной физической памяти, осуществить ввод с клавиатуры строки с выводом введенной строки на экран и получить информацию на экране от системного таймера или в виде мигающего курсора, или в виде количества тиков с момента запуска программы на выполнение, или в виде значения реального времени.

**Код программы:**

```
.386p

descr struc
        lim     dw 0
        base_l  dw 0
        base_m          db 0
        attr_1  db 0
        attr_2  db 0
        base_h  db 0
descr ends

int_descr struc
        offs_l  dw 0
        sel             dw 0
        counter db 0
        attr    db 0
        offs_h  dw 0
int_descr ends

; Защищённый режим
PrMod_seg SEGMENT PARA PUBLIC 'CODE' USE32
ASSUME      CS:PrMod_seg

; Таблица дескрипторов сегметов GDT
GDT     label byte
        gdt_null        descr <>
        gdt_flat_DS     descr <0FFFFh,0,0,92h,11001111b,0>
```

```
        gdt_16b_CS      descr <RM_seg_size-1,0,0,98h,0,0>
        gdt_32b_CS      descr <PrMod_seg_size-1,0,0,98h,01000000b,0>
        gdt_32b_DS      descr <PrMod_seg_size-1,0,0,92h,01000000b,0>
        gdt_32b_SS      descr <stack_l-1,0,0, 92h, 01000000b,0>
gdt_size = $-GDT

gdtr df 0

; Селекторы сегментных дескрипторов
selector_flat_DS    equ   8
selector_16b_CS     equ   16
selector_32b_CS     equ   24
selector_32b_DS     equ   32
selector_32b_SS     equ   40

; Таблица дескрипторов прерываний IDT
IDT     label byte
        trap1 int_descr 13 dup (<,selector_32b_CS,,8fh>)
        trap13 int_descr <0, selector_32b_CS,,8fh> ; Исключение #13
   trap2 int_descr 18 dup (<,selector_32b_CS,,8fh>)
        timer    int_descr <0, selector_32b_CS,0, 8Eh, 0>
        keyboard int_descr <0, selector_32b_CS,0, 8Eh, 0>
idt_size = $-IDT

idtr        df 0
idtr_real dw 3FFh, 0, 0

mask_master     db 0
mask_slave   db 0

escape              db 0
time_counter dd 0

msg1 db 'Real Mode!. To move to Protected Mode press key!$'
msg2 db 'Real Mode again!$'

cur_out_pos         dd 1E0h  ;Позиция печати вводимого текста

; Таблица символов ASCII для перевода из скан кода в код ASCII.
table_ascii         db 0, 0, 49, 50, 51, 52, 53, 54, 55, 56, 57, 48, 45, 61, 0, 0
                                db 81, 87, 69, 82, 84, 89, 85, 73, 79, 80, 91, 93, 0, 0, 65, 83
                                db 68, 70, 71, 72, 74, 75, 76, 59, 39, 96, 0, 92, 90, 88, 67
                                db 86, 66, 78, 77, 44, 46, 47
```

```
print_str macro str
	mov ah,9
	mov dx, str
	int 21h
endm


create_number macro
	local number1
		cmp dl,10
		jl number1
		add dl,'A' - '0' - 10

	number1:
		add dl,'0'
endm


; макрос печати на экран значения регистра EAX через видеобуффер
my_print_eax macro
	local prcyc1
		push ecx
		push dx
		mov ecx,8
		add ebp, 0B8010h

	prcyc1:
		mov dl,al
		and dl,0Fh
		create_number 0
		mov es:[ebp],dl
		ror eax,4
		sub ebp,2
		loop prcyc1

		sub ebp,0B8010h
		pop dx
		pop ecx
endm


; точка входа в 32-битный защищенный режим
PrMod_entry:
```

```
        ; Устанавливаем селекторы в сегментные регистры
        mov     ax,selector_32b_DS
        mov     ds,ax
        mov     ax,selector_flat_DS
        mov     es,ax
        mov     ax,selector_32b_SS
        mov     ebx,stack_l
        mov     ss,ax
        mov     esp,ebx

        ; разрешить прерывания
        sti
        ;считаем количество доступной памяти и печатаем его на экран
        call compute_memory

work:
        test escape, 1
        jz work

go_back:
                cli
                db      0EAh
                dd      offset RealMod_return
                dw      selector_16b_CS

new_timer:
        push eax
        push ebp
        push ecx
        push dx
        mov  eax,time_counter

        push ebp
        ; указываем смещение в видеопамяти относительно начала экрана
        ;(10 символов - 1 байт символа и 1 байт цвета)
        mov ebp, 0
        my_print_eax 0
        pop ebp

        inc eax
        mov time_counter,eax

        pop dx
        pop ecx
```

```
        pop ebp

        mov     al,20h
        out     20h,al
        pop eax

        iretd

;Новый обработчик прерывания клавиатуры для защищенного режима
new_keyboard:
        push eax
        push ebx
        push ebp
        push edx

        in      al, 60h

        cmp     al,1Ch
        jne     not_leave
        mov escape,1
        jmp leav_

not_leave:
        cmp al,80h
        ja leav_
        xor ah,ah
        mov bp,ax
        mov dl,table_ascii[ebp]
        mov ebp,0B8000h
        mov ebx,cur_out_pos  ; Текущая позиция вывода символа
        mov es:[ebp+ebx],dl

        add ebx,2       ; Увеличим текущую позицию вывода текста и сохраним ее
        mov cur_out_pos,ebx

leav_:
        in      al,61h
        or      al,80h
        out     61h,al

        ; Посылаем сигнал EOI
        mov     al,20h
        out     20h,al
```

```
        pop edx
        pop ebp
        pop ebx
        pop     eax
        iretd


;функция подсчета доступной памяти
compute_memory          proc
        push ds
        mov     ax,  selector_flat_DS    ; сегмент на 4 ГБ - все доступное виртуальное АП
        mov     ds,  ax
        mov     ebx, 100001h ; пропускаем первый мегабайт сегмента
        mov     dl,  10101010b

        ; в ECX помещаем количество оставшейся памяти
        ;(до превышения лимита в 4ГБ)
        mov     ecx, 0FFEFFFFEh

; в цикле считаем память
calcloop:
        mov     dh, ds:[ebx]
        mov     ds:[ebx], dl
        cmp     ds:[ebx], dl
        jnz     end_memory
        mov     ds:[ebx], dh
        inc     ebx
        loop calcloop


end_memory:
        pop     ds
        xor     edx, edx
        mov     eax, ebx ;в EBX лежит количество посчитанной памяти в байтах;
        mov     ebx, 100000h
        div     ebx

        push ebp
        ; указываем смещение в видеопамяти относительно начала экрана
        ;(10 символов - 1 байт символа и 1 байт цвета)
        mov ebp,20
        my_print_eax 0
        pop ebp
        ret
```

```
compute_memory          endp

; заглушка для исключений 1-32 (за исключением 13)
common_stub proc
        iretd
common_stub endp

; заглушка для исключения 13
stub13 proc
        pop EAX
        pop EAX
        shr EAX, 16
        iretd
stub13 endp

PrMod_seg_size = $-GDT
PrMod_seg       ENDS


stack_seg       SEGMENT  PARA STACK 'STACK'
        stack_start     db      100h dup(?)
        stack_l = $-stack_start   ;длина стека для инициализации ESP
stack_seg       ENDS




; Реальный режим
RM_seg          SEGMENT PARA PUBLIC 'CODE' USE16
        ASSUME CS:RM_seg, DS:PrMod_seg, SS:stack_seg

start:

                mov   ax,PrMod_seg
                mov   ds,ax

                mov ah, 09h
                mov edx, offset msg1
                int 21h

                push eax
                mov ah,10h
                int 16h
                pop eax
```

```
; очистить экран
mov     ax,3
int     10h

; настраиваем регистр ds на сегмент с защищенном режимом
push PrMod_seg
pop ds

; вычислить базы для всех используемых дескрипторов сегментов
xor     eax,eax
mov     ax,RM_seg
shl     eax,4
mov     word ptr gdt_16b_CS.base_l,ax
shr     eax,16
mov     byte ptr gdt_16b_CS.base_m,al
mov     ax,PrMod_seg
shl     eax,4
push eax
push eax
mov     word ptr gdt_32b_CS.base_l,ax
mov     word ptr gdt_32b_SS.base_l,ax
mov     word ptr gdt_32b_DS.base_l,ax
shr     eax,16
mov     byte ptr gdt_32b_CS.base_m,al
mov     byte ptr gdt_32b_SS.base_m,al
mov     byte ptr gdt_32b_DS.base_m,al

; вычислим линейный адрес GDT
pop eax
add     eax,offset GDT ; в eax будет полный линейный адрес GDT (адрес сегмента + смещение GDT относительно него)
mov     dword ptr gdtr+2,eax
mov word ptr gdtr, gdt_size-1
; загрузим GDT
lgdt    fword ptr gdtr

; аналогично вычислим линейный адрес IDT
pop     eax
add     eax,offset IDT
mov     dword ptr idtr+2,eax
mov word ptr idtr, idt_size-1

;заполним смещение в дескрипторах прерываний
```

```
mov EAX, offset common_stub
mov trap1.offs_l, AX
mov trap2.offs_l, AX
shr EAX, 16
mov trap1.offs_h, AX

mov trap2.offs_h, AX
mov EAX, offset stub13
mov trap13.offs_l, AX
shr EAX, 16
mov trap13.offs_h, AX


mov     eax, offset new_timer
mov     timer.offs_l, ax
shr     eax, 16
mov     timer.offs_h, ax
mov     eax, offset new_keyboard
mov     keyboard.offs_l, ax
shr     eax, 16
mov     keyboard.offs_h, ax

;сохраним маски прерываний контроллеров
in      al, 21h
mov     mask_master, al
in      al, 0A1h
mov     mask_slave, al

;перепрограммируем ведущий контроллер)
mov     al, 11h
out     20h, al
mov     AL, 20h
out     21h, al
mov     al, 4
out     21h, al
mov     al, 1
out     21h, al

; Запретим все прерывания в ведущем контроллере, кроме IRQ0(таймер) и
IRQ1(клавиатура)
mov     al, 0FCh
out     21h, al

;запретим все прерывания в ведомом контроллере
```

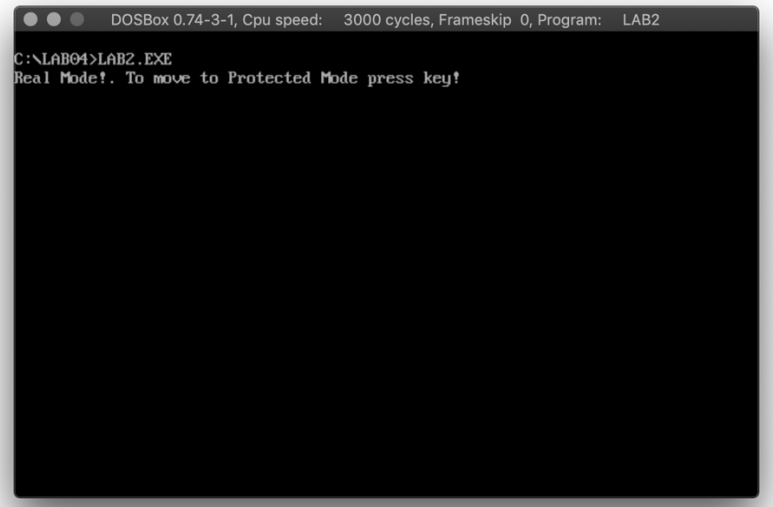
```
        mov     al, 0FFh
        out     0A1h, al

        ; загрузим IDT
        lidt fword ptr idtr

        ; A20 - линия, через которую осуществляется доступ ко всей памяти за
пределами первого мегабайта
        in      al,92h
        or      al,2
        out     92h,al

        ; отключить маскируемые прерывания
        cli
        ; отключим немаскируемые прерывания
        in      al,70h
        or      al,80h
        out     70h,al

        ; перейти в защищенный режим установкой соответствующего бита регистра
CR0
        mov     eax,cr0
        or      al,1
        mov     cr0,eax

        db      66h
        db      0EAh
        dd      offset PrMod_entry
        dw      selector_32b_CS

RealMod_return:
        mov     eax,cr0
        and     al,0FEh
        mov     cr0,eax

        db      0EAh
        dw      $+4
        dw      RM_seg

        ; восстановить регистры для работы в реальном режиме
        mov     ax,PrMod_seg
        mov     ds,ax
        mov     es,ax
        mov     ax,stack_seg
        mov     bx,stack_l
```

```
                mov     ss,ax
                mov     sp,bx

                mov     al, 11h
                out     20h, al
                mov     al, 8
                out     21h, al
                mov     al, 4
                out     21h, al
                mov     al, 1
                out     21h, al

                ;восстанавливаем маски контроллеров прерываний
                mov     al, mask_master
                out     21h, al
                mov     al, mask_slave
                out     0A1h, al

                ; загружаем таблицу дескрипторов прерываний реального режима
                lidt fword ptr idtr_real

                ; разрешаем немаскируемые прерывания
                in      al,70h
                and     al,07FH
                out     70h,al

        ; разрешаем маскируемые прерывания
                sti

                ; очистить экран
                mov     ax,3
                int     10h

                ; печать сообщения о выходе из защищенного режима
                mov ah, 09h
                mov edx, offset msg2
                int 21h

                mov     ah,4Ch
                int     21h

RM_seg_size = $-start
RM_seg          ENDS
END start
```

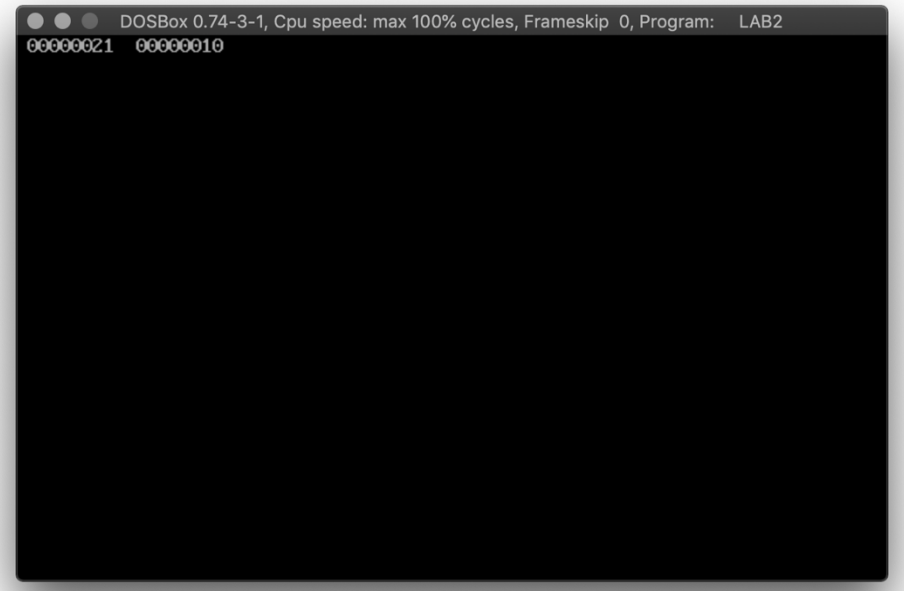
**Демонстрация работы программы**

На первом изображении строчка «Real Mode! To move to Protected Mode press key!» печатается в реальном режиме.
После нажатия на любую кнопку происходит переход в защищённый режим (изображение 2). Слева выводится счётчик времени; справа выводится размер доступной физической памяти (у меня это 16Мб). Затем снова идёт переход в реальный режим (изображение 3).

Изображение 1:




Изображение 2 :




Изображение 3 (возвращение в реальный режим):