



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

(Группа)

А.С.Саркисов

\_\_\_\_\_  
(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 работу в реальном режиме. Для перевода в защищенный режим выполняются необходимые действия. В защищенном режиме программа работает на нулевом уровне привилегий.

### Требование к программе:

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
    limit  dw 0
    base_l dw 0
    base_m db 0
    attr_1 db 0
    attr_2 db 0
    base_h db 0
descr ends
```

```
idescr struc
    offs_l dw 0
    sel    dw 0
    cntr   db 0
    attr   db 0
    offs_h dw 0
idescr ends
```

```
stack32 segment para stack 'STACK'
    stack_start db 100h dup(?)
```

```
stack_size = $-stack_start  
stack32 ends
```

```
data32 segment para 'data'
```

```
gdt_null descr <>  
gdt_code16 descr <code16_size-1,0,0,98h>  
gdt_data4gb descr <0FFFFh,0,0,92h,0CFh>  
gdt_code32 descr <code32_size-1,0,0,98h,40h>  
gdt_data32 descr <data_size-1,0,0,92h,40h>  
gdt_stack32 descr <stack_size-1,0,0,92h,40h>  
gdt_video16 descr <3999,8000h,0Bh,92h>
```

```
gdt_size=$-gdt_null  
pdescr df 0
```

```
code16s=8  
data4gbs=16  
code32s=24  
data32s=32  
stack32s=40  
video16s=48
```

```
idt label byte
```

```
:[0; 12]  
idescr_0_12 idescr 13 dup (<0,code32s,0,8Fh,0>)  
idescr_13 idescr <0,code32s,0,8Fh,0>  
:[14; 31]  
idescr_14_31 idescr 18 dup (<0,code32s,0,8Fh,0>)
```

```
int08 idescr <0,code32s,0,10001110b,0>  
int09 idescr <0,code32s,0,10001110b,0>
```

```
idt_size = $-idt
```

```
ipdescr df 0  
ipdescr16 dw 3FFh, 0, 0
```

```
mask_master db 0  
mask_slave db 0
```

```
asciimap 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
```

```
flag_enter_pr db 0
cnt_time db 0
```

```
syml_pos dd 2 * 80 * 5
```

```
interval=10
```

```
mem_pos=30*2
mem_value_pos=36*2
mb_pos=45*2
cursor_pos=80*2*2+38*2
param=1Eh
```

```
cursor_symb_on=220
cursor_symb_off=223
```

```
rm_msg db 27, '[29;44mNow in Real Mode. ', 27, '[0m$'
pm_msg_wait db 27, '[29;44mAny key to enter protected mode!', 27, '[0m$'
pm_msg_out db 27, '[29;44mNow in Real Mode again! ', 27, '[0m$'
```

```
data_size = $-gdt_null
data32 ends
```

```
code32 segment para public 'code' use32
assume cs:code32, ds:data32, ss:stack32
```

```
pm_start:
mov ax, data32s
mov ds, ax
mov ax, video16s
mov es, ax
mov ax, stack32s
mov ss, ax
mov eax, stack_size
mov esp, eax
```

sti

mov di, mem\_pos  
mov ah, param

mov al, 'M'  
stosw  
mov al, 'e'  
stosw  
mov al, 'm'  
stosw  
mov al, 'o'  
stosw  
mov al, 'r'  
stosw  
mov al, 'y'  
stosw  
mov al, ':'  
stosw

call count\_memory

process:  
 test flag\_enter\_pr, 1  
 jz process

cli

db 0EAh  
dd offset return\_rm  
dw code16s

except\_1 proc  
 iret  
except\_1 endp

except\_13 proc uses eax  
 pop eax  
 iret

```
except_13 endp
```

```
new_int08 proc uses eax
```

```
    mov edi, cursor_pos
```

```
    cmp cnt_time, interval
```

```
    je X
```

```
    cmp cnt_time, interval*2
```

```
    jne skip
```

```
    mov al, cursor_symb_off
```

```
    mov cnt_time, 0
```

```
    jmp pr
```

```
X:
```

```
    mov al, cursor_symb_on
```

```
pr:
```

```
    mov ah, param
```

```
    stosw
```

```
skip:
```

```
    mov al, cnt_time
```

```
    inc al
```

```
    mov cnt_time, al
```

```
    mov al, 20h
```

```
    out 20h, al
```

```
    iretd
```

```
new_int08 endp
```

```
new_int09 proc uses eax ebx edx
```

```
    in al, 60h
```

```
    cmp al, 1Ch
```

```
    jne print_value
```

```
    or flag_enter_pr, 1
```

```
    jmp allow_handle_keyboard
```

```
print_value:
```

```
    cmp al, 80h
```

```
    ja allow_handle_keyboard
```

```

xor ah, ah

xor ebx, ebx
mov bx, ax

mov dh, param
mov dl, asciimap[ebx]
mov ebx, syml_pos
mov es:[ebx], dx

add ebx, 2
mov syml_pos, ebx

allow_handle_keyboard:
in al, 61h
or al, 80h
out 61h, al
and al, 7Fh
out 61h, al

mov al, 20h
out 20h, al

iretd
new_int09 endp

count_memory proc uses ds eax ebx
mov ax, data4gbs
mov ds, ax

mov ebx, 100001h
mov dl, 0AEh

mov ecx, 0FFEFFFEh

iterate_through_memory:
mov dh, ds:[ebx]

mov ds:[ebx], dl
cmp ds:[ebx], dl

```

```

        jnz print_memory_counter

        mov ds:[ebx], dh
        inc ebx
    loop iterate_through_memory

print_memory_counter:
    mov eax, ebx
    xor edx, edx

    mov ebx, 100000h
    div ebx

    ; change place
    mov ebx, mem_value_pos
    call print_eax

    ; change place
    mov ah, param
    mov ebx, mb_pos
    mov al, 'M'
    mov es:[ebx], ax

    mov ebx, mb_pos + 2
    mov al, 'b'
    mov es:[ebx], ax
    ret
count_memory endp

```

```

print_eax proc uses ecx ebx edx
    add ebx, 10h
    mov ecx, 8
    mov dh, param

```

```

print_symbol:
    mov dl, al
    and dl, 0Fh

    cmp dl, 10
    jl add_zero_sym

```



```
add dl, 'A' - '0' - 10
```

```
add_zero_sym:
```

```
add dl, '0'
```

```
mov es:[ebx], dx
```

```
ror eax, 4
```

```
sub ebx, 2
```

```
loop print_symbol
```

```
ret
```

```
print_eax endp
```

```
code32_size = $-pm_start
```

```
code32 ends
```

```
code16 segment para public 'CODE' use16
```

```
assume cs:code16, ds:data32, ss: stack32
```

```
start:
```

```
mov ax, data32
```

```
mov ds, ax
```

```
mov ah, 09h
```

```
lea dx, rm_msg
```

```
int 21h
```

```
xor dx, dx
```

```
mov ah, 2
```

```
mov dl, 13
```

```
int 21h
```

```
mov dl, 10
```

```
int 21h
```

```
mov ah, 09h
```

```
lea dx, pm_msg_wait
```

```
int 21h
```

```
xor dx, dx
```

```
mov ah, 2
```

```
mov dl, 13
```

```
int 21h
```

```
mov dl, 10
```

```
int 21h
```

```
;key waiting  
mov ah, 10h  
int 16h
```

```
; clear screen  
mov ax, 3  
int 10h
```

```
xor eax, eax
```

```
mov ax, code16  
shl eax, 4  
mov word ptr gdt_code16.base_l, ax  
shr eax, 16  
mov byte ptr gdt_code16.base_m, al  
mov byte ptr gdt_code16.base_h, ah
```

```
mov ax, code32  
shl eax, 4  
mov word ptr gdt_code32.base_l, ax  
shr eax, 16  
mov byte ptr gdt_code32.base_m, al  
mov byte ptr gdt_code32.base_h, ah
```

```
mov ax, data32  
shl eax, 4  
mov word ptr gdt_data32.base_l, ax  
shr eax, 16  
mov byte ptr gdt_data32.base_m, al  
mov byte ptr gdt_data32.base_h, ah
```

```
mov ax, stack32  
shl eax, 4  
mov word ptr gdt_stack32.base_l, ax  
shr eax, 16  
mov byte ptr gdt_stack32.base_m, al  
mov byte ptr gdt_stack32.base_h, ah
```

```
mov ax, data32  
shl eax, 4
```

```
add eax, offset gdt_null
```

```
mov dword ptr pdescr+2, eax  
mov word ptr pdescr, gdt_size-1  
lgdt fword ptr pdescr
```

```
lea eax, es:except_1  
mov idescr_0_12.off_1, ax  
shr eax, 16  
mov idescr_0_12.off_h, ax
```

```
lea eax, es:except_13  
mov idescr_13.off_1, ax  
shr eax, 16  
mov idescr_13.off_h, ax
```

```
lea eax, es:except_1  
mov idescr_14_31.off_1, ax  
shr eax, 16  
mov idescr_14_31.off_h, ax
```

```
lea eax, es:new_int08  
mov int08.off_1, ax  
shr eax, 16  
mov int08.off_h, ax
```

```
lea eax, es:new_int09  
mov int09.off_1, ax  
shr eax, 16  
mov int09.off_h, ax
```

```
mov ax, data32  
shl eax, 4  
add eax, offset idt
```

```
mov dword ptr ipdescr + 2, eax  
mov word ptr ipdescr, idt_size-1
```

```
; сохранение масок  
in al, 21h
```

```
mov mask_master, al
in al, 0A1h
mov mask_slave, al
```

```
; перепрограммирование ведущего контроллера
```

```
mov al, 11h
out 20h, al
mov al, 32
out 21h, al
mov al, 4
out 21h, al
mov al, 1
out 21h, al
```

```
; маска для ведущего контроллера
```

```
mov al, 0FCh
out 21h, al
```

```
; маска для ведомого контроллера (запрещаем прерывания)
```

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

```
lidt fword ptr ipdescr
```

```
; открытие линии A20
```

```
in al, 92h
or al, 2
out 92h, al
```

```
cli
```

```
mov eax, cr0
or eax, 1
mov cr0, eax
```

```
db 66h
db 0EAh
dd offset pm_start
dw code32s
```

```
return_rm:
```

```
mov eax, cr0
and al, 0FEh
mov cr0, eax
```

```
db 0EAh
dw offset go
dw code16
```

go:

```
mov ax, data32
mov ds, ax
mov ax, code32
mov es, ax
mov ax, stack32
mov ss, ax
mov ax, stack_size
mov sp, ax
```

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

```
mov al, mask_master
out 21h, al
mov al, mask_slave
out 0A1h, al
```

```
lidt fword ptr ipdescr16
```

```
;A20
in al, 70h
and al, 7Fh
out 70h, al
```

```
sti
```

```
;clear screen
```

```
mov ax, 3  
int 10h
```

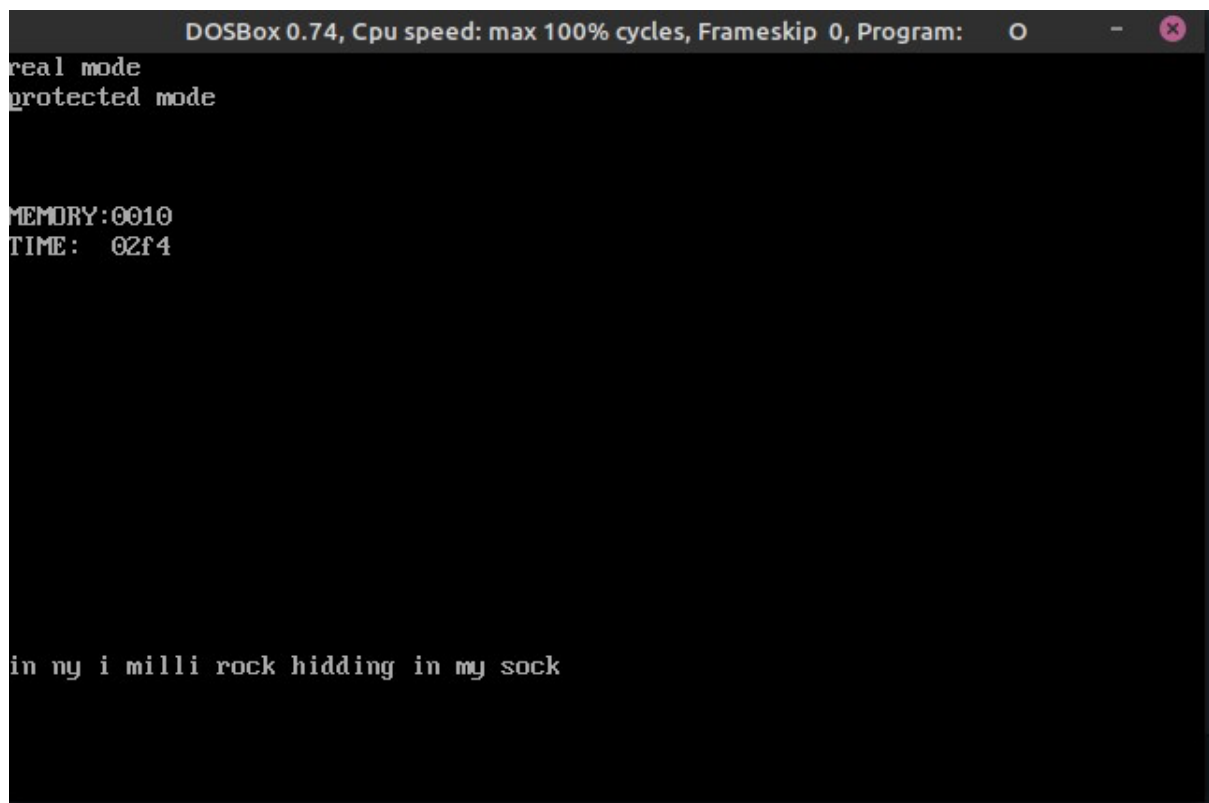
```
mov ah, 09h  
lea dx, pm_msg_out  
int 21h  
xor dx, dx  
mov ah, 2  
mov dl, 13  
int 21h  
mov dl, 10  
int 21h
```

```
mov ax, 4C00h  
int 21h
```

```
code16_size = $-start  
code16 ends
```

```
end start
```

## Демонстрация работы программы



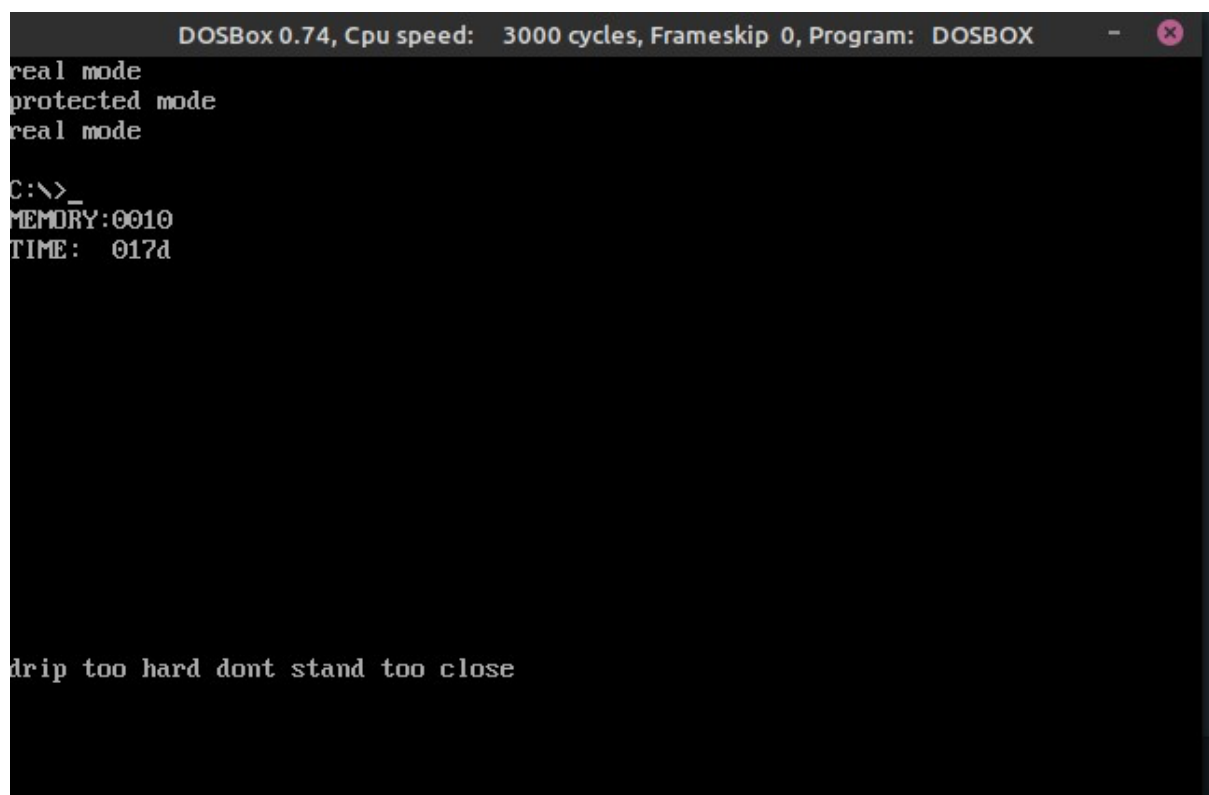
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: O

```
real mode
protected mode

MEMORY:0010
TIME: 02f4

in ny i milli rock hidding in my sock
```

MEMORY:0010 показывает объем доступной физической памяти (32Мб). По умолчанию DOSBOX выделено 16Мб. ESC переводит программу обратно в реальный режим.



DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
real mode
protected mode
real mode

C:\>_
MEMORY:0010
TIME: 017d

drip too hard dont stand too close
```