**Код программы**

.386p

descr struc; Структура для описания декскриптора сегмента в таблице глобальных дескрипторов GDT

limit dw 0 ; Граница (биты 0..15) - размер сегмента в байтах

base\_l dw 0 ; Младшие 16 битов адресной базы. - базовый адрес задаётся в виртуальном адресном пространстве

base\_m db 0 ; Следующие 8 битов адресной базы.

attr\_1 db 0 ; Флаги/атрибуты доступа, определяющие в каком кольце защиты

arrt\_2 db 0 ; можно использовать этот сегмент.

base\_h db 0 ; Последние 8 битов адресной базы.

descr ends

intr struc

offs\_l dw 0

sel dw 0

rsrv db 0

attr db 0

offs\_h dw 0

intr ends

pm\_seg segment para public 'code' use32

assume cs:pm\_seg

***gdt label byte***

***gdt\_null descr <>***

***gdt\_data descr <0FFFFh,0,0,92h,0CFh,0>***

***gdt\_code16 descr <rm\_seg\_size-1,0,0,98h,0,0>***

***gdt\_code32 descr <pm\_seg\_size-1,0,0,98h,0CFh,0>***

***gdt\_data32 descr <pm\_seg\_size-1,0,0,92h,0CFh,0>***

***gdt\_stack32 descr <stack\_size-1,0,0,92h,0CFh,0>***

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

***gdt\_size=$-gdt***

gdtr dw gdt\_size-1

dd ?

; Селекторы сегментов

sel\_data equ 8

sel\_code16 equ 16

sel\_code32 equ 24

sel\_data32 equ 32

sel\_stack32 equ 40

selector\_screen equ 48

***idt label byte***

***trap1 intr 13 dup (<0,sel\_code32,0,8Fh,0>)***

***trap13 intr <0,sel\_code32,0,8Fh,0>***

***trap2 intr 18 dup (<0,sel\_code32,0,8Fh,0>)***

***int\_time intr <0,sel\_code32,0,8Eh,0>***

***int\_keyboard intr <0,sel\_code32,0,8Eh,0>***

***idt\_size=$-idt***

idtr dw idt\_size-1

dd ?

rm\_idtr dw 3FFh,0,0

hex db 'h'

hex\_len=$-hex

mb db 'MB'

mb\_len=$-mb

hello\_msg db 'DOS is in real mode now.$'

pm\_msg db 'DOS switched to protected mode.'

pm\_msg\_len=$-pm\_msg

tt\_msg db 'Timer ticks: '

tt\_msg\_len=$-tt\_msg

am\_msg db 'Available memory: '

am\_msg\_len=$-am\_msg

esc\_from\_pr db 'Press ESC to switch to real mode...'

esc\_from\_pr\_len=$-esc\_from\_pr

ret\_to\_rm\_msg db 'DOS switched to real mode.$'

scan2ascii 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

attr1 db 0Fh

attr2 db 0Fh

screen\_addr dd 640

timer dd 0

master db 0

slave db 0

; Макрос вывода строки в видеобуфер

print\_str macro msg,len,offset

local print

push ecx

push ebp

push eax

push ebx

push es

mov ecx, len

mov ah, attr2

mov bx, selector\_screen

mov es, bx

mov ebx, offset

xor esi, esi

print:

mov al,byte ptr msg[esi]

mov es:[ebx],ax

add ebx,2

inc esi

loop print

pop es

pop ebx

pop eax

pop ebp

pop ecx

endm

; Макрос отправки сигнал EOI контроллеру прерываний

send\_eoi macro

mov al,20h

out 20h,al

endm

pm\_start:

; Установить сегменты защищенного режима

mov ax,sel\_data

mov ds,ax

mov es,ax

mov ax,sel\_stack32

mov ebx,stack\_size

mov ss,ax

mov esp,ebx

; Разрешить маскируемые прерывания

sti

; Вывести справочную информацию в видеобуфер

print\_str pm\_msg,pm\_msg\_len,360

print\_str tt\_msg,tt\_msg\_len,520

print\_str am\_msg,am\_msg\_len,5\*160+40

print\_str esc\_from\_pr,esc\_from\_pr\_len,6\*160+40

call available\_memory

jmp $

; Обработчик исключения общей защиты

exc13 proc

pop eax

iret

exc13 endp

; Обработчик остальных исключений

dummy\_exc proc

iret

dummy\_exc endp

; Обработчик прерывания от системного таймера

int\_time\_handler:

push eax

push ebp

push ecx

push dx

push es

push ebx

; Загрузить счетчик

mov eax,timer

; Вывести счетчик в видеобуфер

mov dh,attr2

mov bx, selector\_screen

mov es, bx

mov ecx, 8

mov ebp, 530+2\*(tt\_msg\_len)

main\_loop:

mov dl,al

and dl,0Fh

cmp dl,10

jl less\_than\_10

sub dl,10

add dl,'A'

jmp print

less\_than\_10:

add dl,'0'

print:

mov es:[ebp],dx

ror eax,4

sub ebp,2

loop main\_loop

; Инкрементировать и сохранить счетчик

inc eax

mov timer,eax

send\_eoi

pop ebx

pop es

pop dx

pop ecx

pop ebp

pop eax

iretd

; Обработчик прерывания от клавиатуры

int\_keyboard\_handler:

push eax

push ebx

push es

push ds

; Получить из порта клавиатуры скан-код нажатой клавиши

in al,60h

; Нажата клавиша ESC

cmp al,01h

je esc\_pressed

; Нажата необслуживаемая клавиша

cmp al,39h

ja skip\_translate

; Преобразовать скан-код в ASCII

mov bx,sel\_data32

mov ds,bx

mov ebx,offset scan2ascii

xlatb

mov bx,sel\_data

mov es,bx

mov ebx,screen\_addr

mov ah,attr2

mov cx, selector\_screen

mov es, cx

; Нажата клавиша Backspace

cmp al,8

je bs\_pressed

; Вывести символ на экран

mov es:[ebx],al

add dword ptr screen\_addr,2

jmp skip\_translate

bs\_pressed:

; Удалить символ

mov al,' '

sub ebx,2

mov es:[ebx+0B8000h],al

mov screen\_addr,ebx

skip\_translate:

; Разрешить работу клавиатуры

in al,61h

or al,80h

out 61h,al

send\_eoi

pop ds

pop es

pop ebx

pop eax

iretd

esc\_pressed:

; Разрешить работу клавиатуры

in al,61h

or al,80h

out 61h,al

send\_eoi

pop ds

pop es

pop ebx

pop eax

; Запретить маскируемые прерывания

cli

; Вернуться в реальный режим

db 0EAh

dd offset rm\_return

dw sel\_code16

; Процедура определения доступного объема оперативной памяти

available\_memory proc

push ds

mov ax,sel\_data

mov ds,ax

; Пропустить первый мегабайт памяти

mov ebx,100001h

; Установить проверочный байт

mov dl,0FFh

; Установить максимальный объем оставшейся оперативной памяти

mov ecx,0FFEFFFFFh

check:

; Проверка сигнатуры

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

; Разделить на мегабайт

mov ebx,100000h

div ebx

push ecx

push dx

push ebp

; Вывести объем памяти на экран

mov dh,attr2

mov bx, selector\_screen

mov es, bx

mov ecx, 8

mov ebp,5\*160+2\*(am\_msg\_len+7)+40

mov dh,attr2

cycle:

mov dl,al

and dl,0Fh

cmp dl,10

jl number

sub dl,10

add dl,'A'

jmp print\_m

number:

add dl,'0'

print\_m:

mov es:[ebp],dx

ror eax,4

sub ebp,2

loop cycle

sub ebp,0B8000h

pop ebp

pop dx

pop ecx

ret

available\_memory endp

pm\_seg\_size=$-gdt

pm\_seg ends

rm\_seg segment para public 'code' use16

assume cs:rm\_seg,ds:pm\_seg,ss:s\_seg

; Макрос очистки экрана

cls macro

mov ax,3

int 10h

endm

; Макрос печати строки

print\_str macro msg

mov ah,9

mov edx,offset msg

int 21h

endm

rm\_start:

mov ax,pm\_seg

mov ds,ax

cls

mov AX, 0B800h

mov ES, AX

mov DI, 200

mov cx, 24

mov ebx, offset hello\_msg

mov ah, attr1

mov al, byte ptr [ebx]

screen0:

stosw

inc bx

mov al, byte ptr [ebx]

loop screen0

; Вычислить базы для всех используемых дескрипторов сегментов

xor eax,eax

mov ax,rm\_seg

shl eax,4

mov word ptr gdt\_code16+2,ax

shr eax,16

mov byte ptr gdt\_code16+4,al

mov ax,pm\_seg

shl eax,4

push eax

push eax

mov word ptr gdt\_code32+2,ax

mov word ptr gdt\_stack32+2,ax

mov word ptr gdt\_data32+2,ax

shr eax,16

mov byte ptr gdt\_code32+4,al

mov byte ptr gdt\_stack32+4,al

mov byte ptr gdt\_data32+4,al

; Вычислить линейный адрес GDT

pop eax

add eax,offset GDT

mov dword ptr gdtr+2,eax

mov word ptr gdtr,gdt\_size-1

; Установить регистр GDTR

lgdt fword ptr gdtr

; Вычислить линейный адрес IDT

pop eax

add eax,offset idt

mov dword ptr idtr+2,eax

mov word ptr idtr,idt\_size-1

; Заполнить смещения в дескрипторах прерываний

mov eax,offset dummy\_exc

mov trap1.offs\_l,ax

shr eax,16

mov trap1.offs\_h,ax

mov eax,offset exc13

mov trap13.offs\_l,ax

shr eax,16

mov trap13.offs\_h,ax

mov eax,offset dummy\_exc

mov trap2.offs\_l,ax

shr eax,16

mov trap2.offs\_h,ax

mov eax,offset int\_time\_handler

mov int\_time.offs\_l,ax

shr eax,16

mov int\_time.offs\_h,ax

mov eax,offset int\_keyboard\_handler

mov int\_keyboard.offs\_l,ax

shr eax,16

mov int\_keyboard.offs\_h,ax

; Cохранить маски ведущего и ведомого контроллеров прерываний

in al,21h

mov master,al

in al,0A1h

mov slave,al

; Перепрограммировать ведущий контроллер прерываний

mov dx,20h

mov al,11h

out dx,al

inc dx

mov al,20h

out dx,al

mov al,4

out dx,al

mov al,1

out dx, al

; Запретить все прерывания в ведущем контроллере, кроме IRQ0 и IRQ1

mov al,11111100b

out dx,al

; Запретить все прерывания в ведомом контроллере

mov dx,0A1h

mov al,0FFh

out dx,al

; Открыть линию А20

mov al,0D1h

out 64h,al

mov al,0DFh

out 60h,al

; Отключить маскируемые и немаскируемые прерывания

cli

in al,70h

or al,80h

out 70h,al

; Загрузить регистр IDTR

lidt fword ptr idtr

; Перейти в защищенный режим установкой соответствующего бита регистра CR0

mov eax,cr0

or al,1

mov cr0,eax

; Перейти в сегмент кода защищенного режима

db 66h

db 0EAh

dd offset pm\_start

dw sel\_code32

rm\_return:

; Перейти в реальный режим сбросом соответствующего бита регистра CR0

mov eax,cr0

and al,0FEh

mov cr0,eax

; Сбросить очередь и загрузить CS

db 0EAh

dw $+4

dw rm\_seg

; Восстановить регистры для работы в реальном режиме

mov ax,pm\_seg

mov ds,ax

mov es,ax

mov ax,s\_seg

mov ss,ax

mov ax,stack\_size

mov sp,ax

; Инициализировать контроллер 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

; Восстановить маски контроллеров прерываний

mov al,master

out 21h,al

mov al,slave

out 0A1h,al

; Загрузить таблицу дескрипторов прерываний реального режима

lidt fword ptr rm\_idtr

; Закрыть линию А20

mov al,0D1h

out 64h,al

mov al,0DDh

out 60h,al

; Разрешить немаскируемые и маскируемые прерывания

in al,70h

and al,07FH

out 70h,al

sti

;cls

mov AX, 0B800h

mov ES, AX

mov DI, 7\*160+40

mov cx, 26

mov ebx, offset ret\_to\_rm\_msg

mov ah, attr1

mov al, byte ptr [ebx]

screen01:

stosw

inc bx

mov al, byte ptr [ebx]

loop screen01

mov ah,4Ch

int 21h

rm\_seg\_size = $-rm\_start

rm\_seg ends

s\_seg segment para stack 'stack'

stack\_start db 100h dup(?)

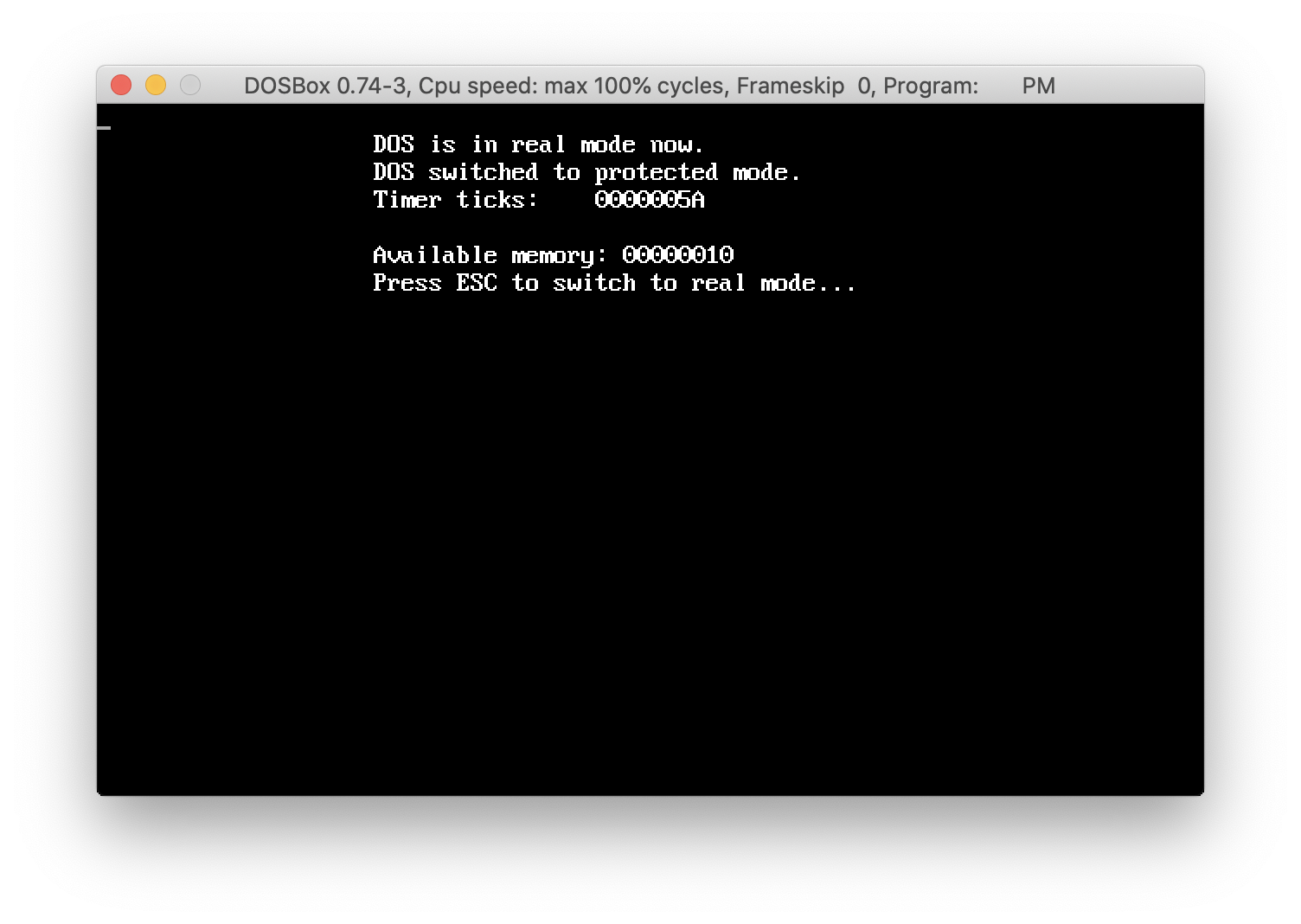
stack\_size=$-stack\_start

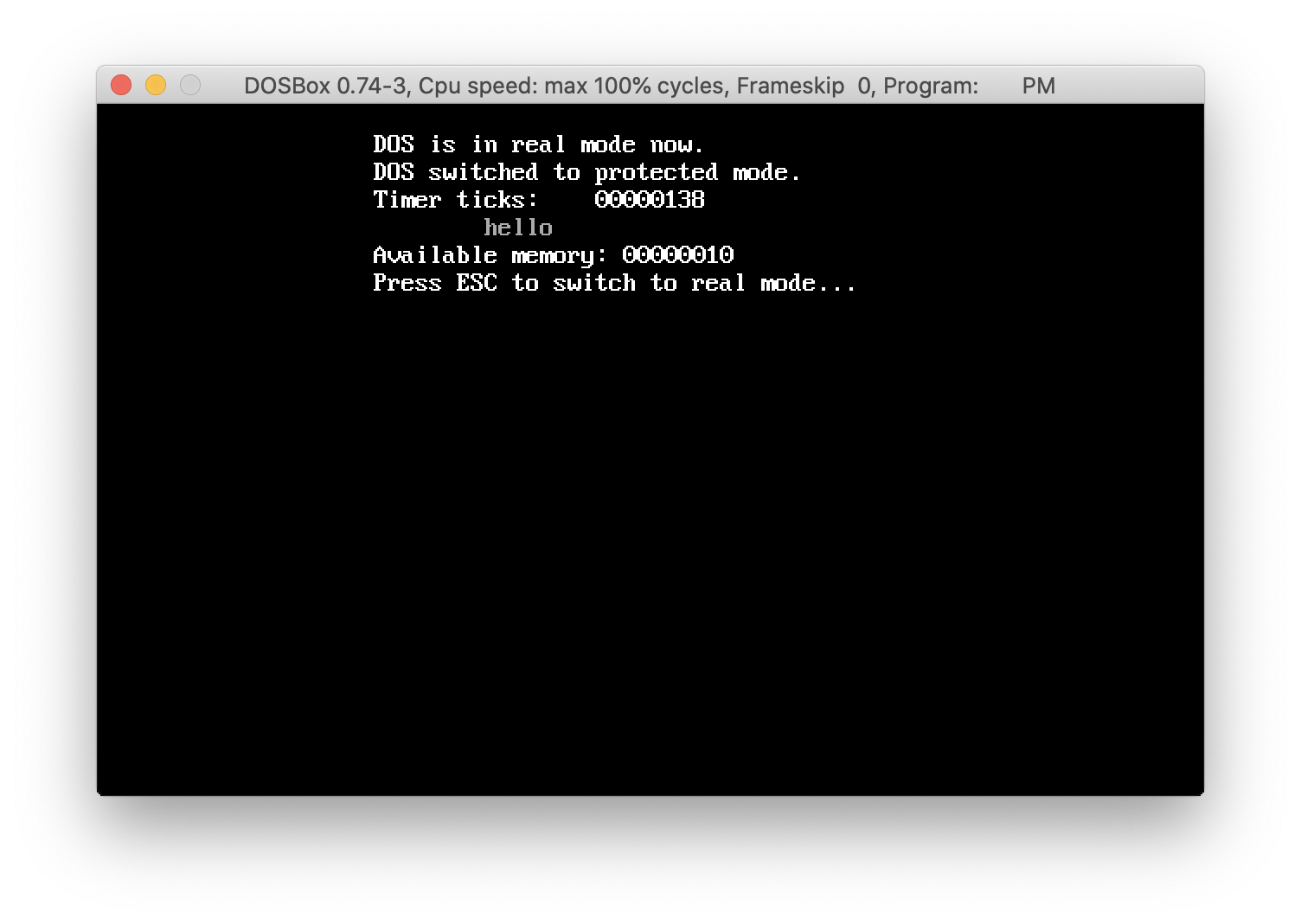
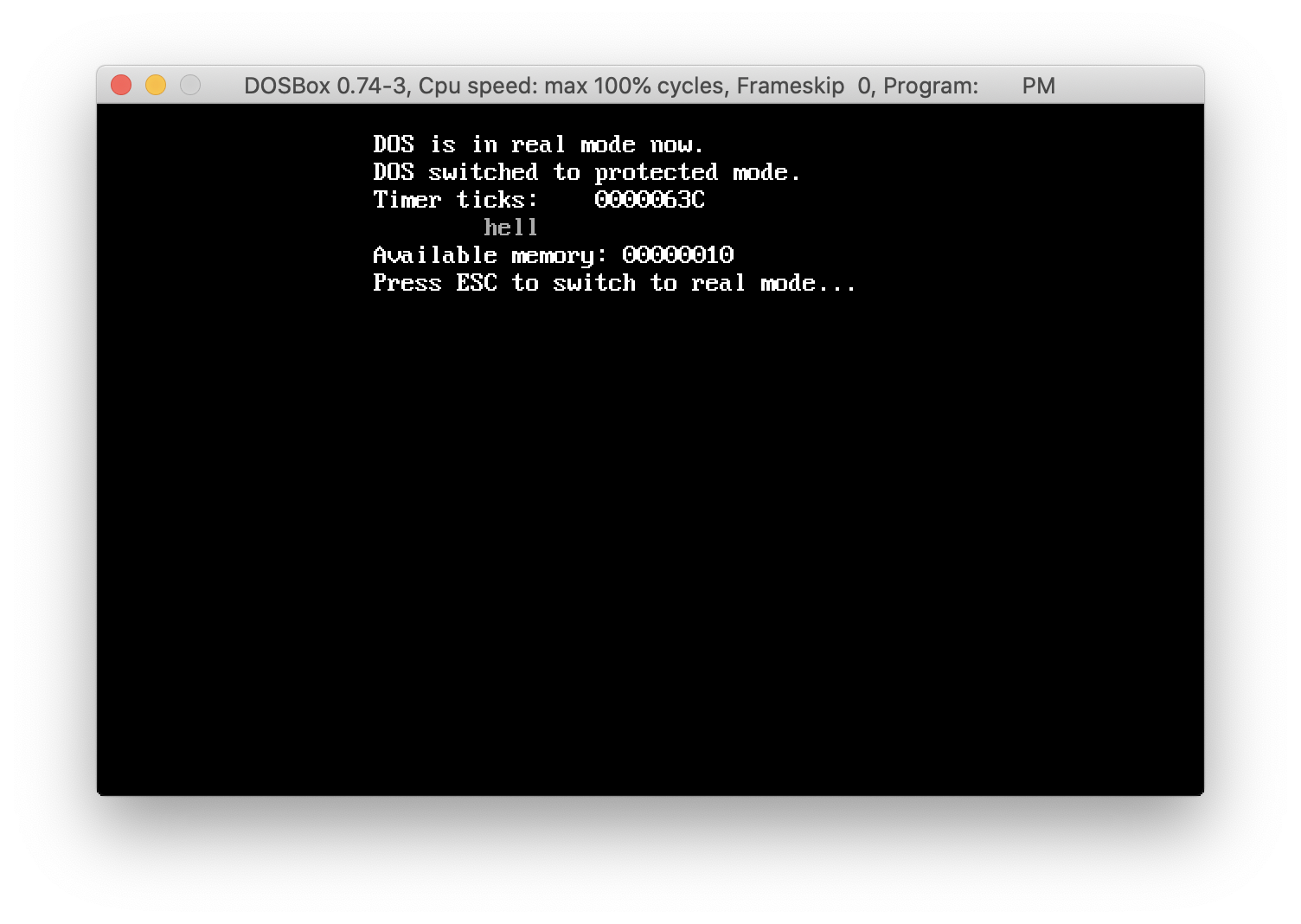
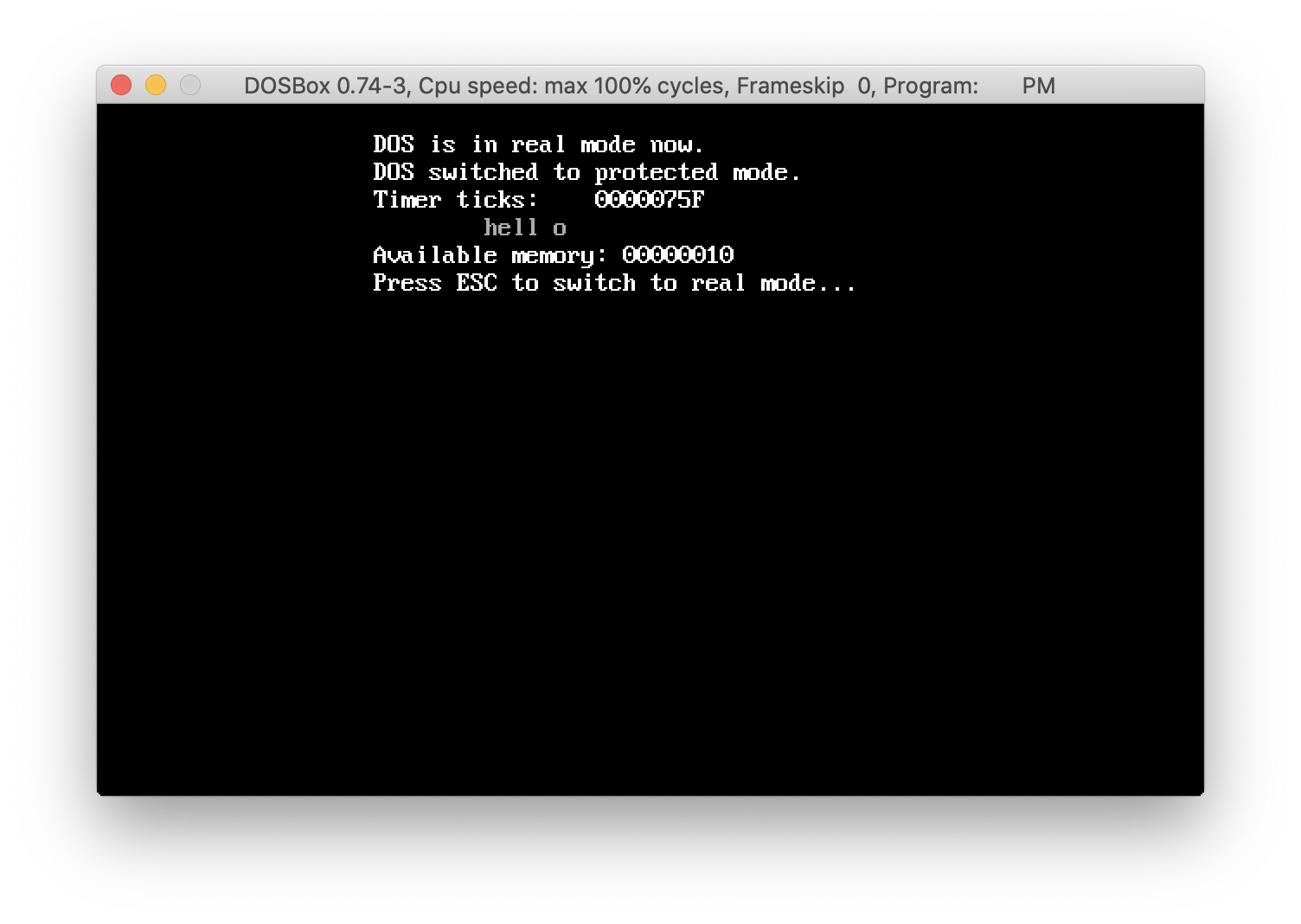
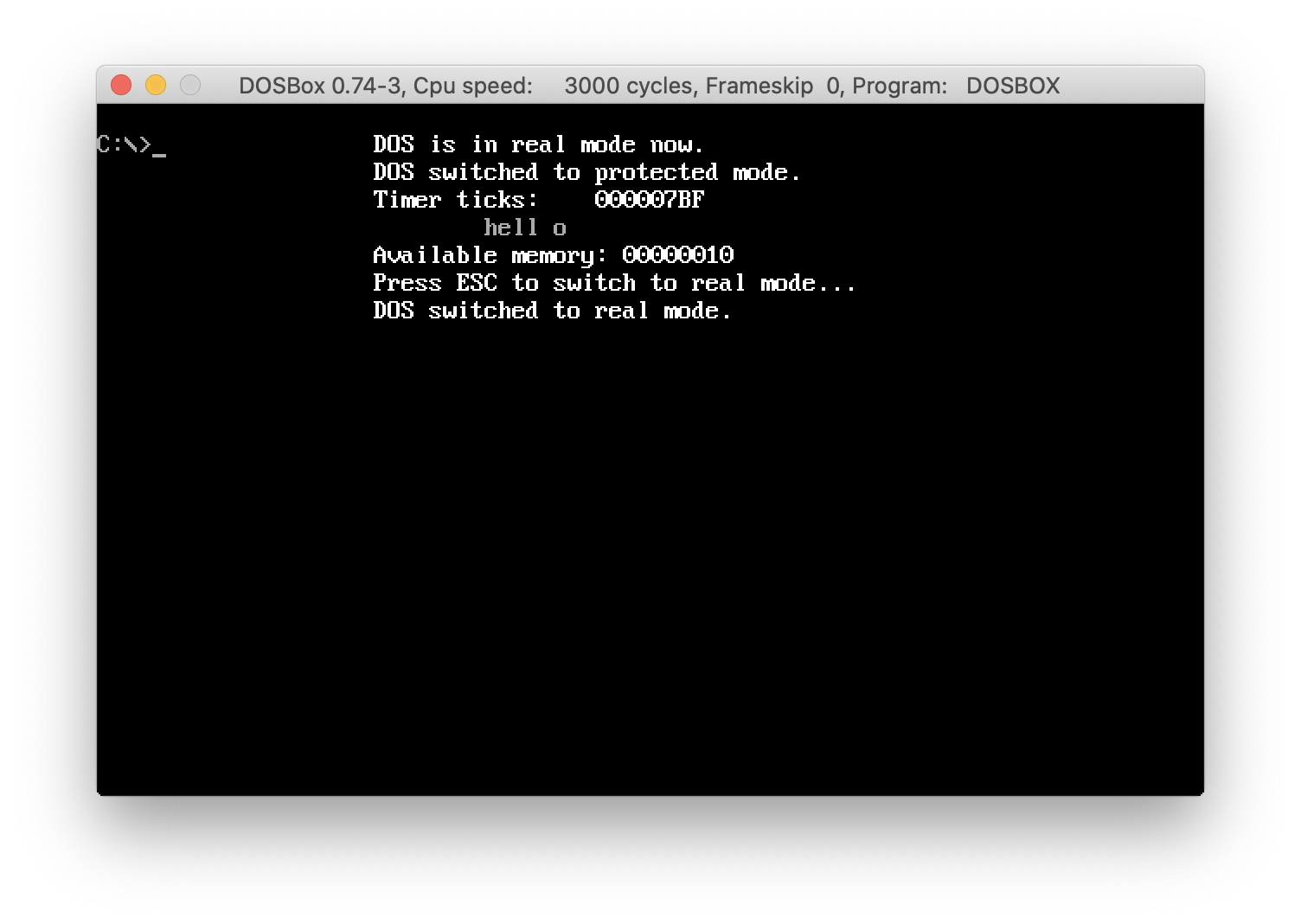
s\_seg ends

end rm\_start

**Демонстрация работы программы**

1. Запускаем программу и осуществляем переход в защищённый режим. Количество выделенной памяти выводится на экран.



1. Проверяем обработчик прерываний от клавиатуры.
2. Клавиша “Backspace” также работает.
3. Клавиша “Space” также работает.
4. Переходим обратно в реальный режим
5. Система продолжает нормально работать