МГТУ им. Баумана

Лабораторная работа №2

По курсу: "Операционные системы"

Защищенный режим.

Работу выполнил: Мокеев Даниил, ИУ7-56

Преподаватель: Рязанова Н.Ю.

Москва, 2020

1 | Листинг кода алгоритмов

```
.386p

descr struc
limit  dw 0
base_l dw 0
base_m db 0
attr_1 db 0
arrt_2 db 0
base_h db 0
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_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

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 3Fh
attr2           db 4Fh
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   ebp
mov    ecx,len
mov    ebp,0B8000h
add    ebp,offset
xor    esi,esi
mov    ah,attr2
print:
mov    al,byte ptr msg[esi]
mov    es:[ebp],ax
add    ebp,2
inc    esi
loop   print
pop    ebp
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

;Загрузить счетчик
mov     eax,timer

;Вывести счетчик в видеобуфер
mov     ebp,0B8000h
mov     ecx,8
add     ebp,530+2*(tt_msg_len)
```

```
mov     dh,attr2
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     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

;Нажата клавиша Backspace
cmp     al,8
je      bs_pressed

;Вывести символ на экран
mov     es:[ebx+0B8000h],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     ebp,0B8000h
mov     ecx,8
add     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

;Сохранить 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

;Загрузить регистр IDTR
lidt    fword ptr idtr

;Открыть линию A20
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

;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 A20
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
```