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
    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

intr struc
    offs_l          dw 0
    sel                 dw 0
    cntr    db 0
    attr        db 0
    offs_h          dw 0
intr ends

stack_seg segment  para stack 'STACK'
    stack_start         db        100h dup(?)
    stack_size = $-stack_start
stack_seg         ENDS

; сегмент данных
data_seg segment para 'DATA'

    gdt_null  descr <>
    gdt_CS_16bit descr <rm_code_size-1, 0, 0, 10011000b, 00000000b, 0>
    gdt_DS_16bit descr <0FFFFh, 0, 0, 10010010b, 10001111b, 0>
    gdt_CS_32bit descr <pm_code_size-1, 0, 0, 10011000b, 01000000b, 0>
    gdt_DS_32bit descr <data_size-1, 0, 0, 10010010b, 01000000b, 0>
    gdt_SS_32bit descr <stack_size-1, 0, 0, 10010110b, 01000000b, 0>
    gdt_VB_32bit descr <3999, 8000h, 0Bh, 10010010b, 01000000b, 0>

    gdt_size = $-gdt_null ; размер таблицы  GDT
    gdtr        df 0
```

```
    sel_CS_16bit    equ    8
    sel_DS_16bit    equ   16
    sel_CS_32bit    equ   24
    sel_DS_32bit    equ   32
    sel_SS_32bit    equ   40
    sel_videobuffer equ   48


    IDT         label byte

        trap_f intr 12 dup (<0, sel_CS_32bit, 0, 10001111b, 0>)
        trap_13 intr <0, sel_CS_32bit, 0, 10001111b, 0>
        trap_s intr 19 dup (<0, sel_CS_32bit, 0, 10001111b, 0>)

    int08 intr <0, sel_CS_32bit, 0, 10001110b, 0>
    int09 intr        <0, sel_CS_32bit, 0, 10001110b, 0>

    idt_size = $-IDT

    idtr df 0
    idtr_backup dw        3FFh, 0, 0

    mask_master        db 0
    mask_slave        db 0

        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

    flag_enter_pr            db 0
    cnt_time                dd 0
    syml_pos     dd 2 * (80 * 10)

    ; far_jump на сегмент, смещение
far_jump macro label, segment
    db        0EAh
    dd        offset label
    dw        segment
```

```
endm

; печать строки на экран
print_str macro str
    mov ah, 09h
    lea dx, str
    int 21h
        xor dx, dx
    mov ah, 2
    mov dl, 13
    int 21h
    mov dl, 10
    int 21h
endm

; ожидание ввода символа с клавиатуры
wait_key macro
    push eax
    mov ah, 10h
    int 16h
    pop eax
endm

; очистить экран
clear_screen macro
    mov         ax, 3
    int         10h
endm

; загружаем дескриптор в таблицу
load_gdt macro gdt_desc
    shl eax, 4                          ; eax - линейный базовый адрес (*2^4 = 16)
    ;(в eax был seg => он выравнен по параграфу => линейный адрес seg * 16)
    mov word ptr gdt_desc.base_l, ax    ; загрузка младшей часть базы
    shr eax, 16                         ; старшую половину eax в ax
    mov byte ptr gdt_desc.base_m, al    ; загрузка средней часть базы
    mov byte ptr gdt_desc.base_h, ah    ; загрузка старшей часть базы
endm
```

```
; загружает в регистр gdtr лин баз адрес gdt и ее размер
init_gdtr macro reg                  ; в reg полный линейный адрес GDT
    mov         dword ptr gdtr + 2, reg            ; кладём полный линейный адрес
    ; в старшие 4 байта переменной gdtr
    mov word ptr  gdtr, gdt_size-1         ; в младшие 2 байта заносим размер gdt,
    ; из-за определения gdt_size (через $) настоящий размер на 1 байт меньше
    lgdt fword ptr gdtr                       ; загрузим GDT
endm

load_idt macro idt_desc
    mov         idt_desc.offs_l, ax
    shr         eax, 16
    mov         idt_desc.offs_h, ax
endm

init_idtr macro reg
    mov          dword ptr idtr + 2, reg
    mov  word ptr  idtr, idt_size-1         idt
endm

;перепрограммируем контроллер
set_interrupt_base macro base
        mov         al, 11h
        out         20h, al

        ; отправляем новый базовый линейный адрес
        mov         al, base
        out         21h, al

        mov         al, 4
        out         21h, al

        mov         al, 1
        out         21h, al
endm

mem_str macro
    mov dI, 0
```

```
    mov ah, 00000110b
    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
endm


    ; выводимые сообщения
    msg_in_rm   db 27, '[32;20mNow in Real Mode. ', 27, '[0m$'
    msg_move_pm db 27, '[32;20mTo enter Protected Mode press any key!', 27, '[0m$'
    msg_out_pm  db 27, '[32;20mNow in Real Mode again! ', 27, '[0m$'

    data_size = $-gdt_null
data_seg ends

pm_seg segment para public 'CODE' use32
    assume cs:pm_seg, ds:data_seg, ss:stack_seg

pm_start:
    ; в регистры сегмента загружаем селекторы
    mov         ax, sel_DS_32bit
    mov         ds, ax
    mov         ax, sel_videobuffer
    mov         es, ax
    mov         ax, sel_SS_32bit
    mov         ss, ax
    mov         eax, stack_size
    mov         esp, eax
```

```
    sti ; разрешить прерывания, запрещенные в реальном режиме
    mem_str
    call count_memory

        ; Возвращение в реальный режим происходит по нажатию
        ; клавиши 'enter' - это будет обработано в коде обработчика прервания
        ; чтобы программа не завершалась до этого момента, нужен бесконечный цикл
    proccess:
        test flag_enter_pr, 1
        jz        proccess

        ; запрещаем прерывания
        ; немаскируемые уже запрещены
        cli ; сброс флага прерывания IF = 0
        far_jump return_rm, sel_CS_16bit

dummy_exc proc
    iret
dummy_exc endp

exc13 proc
    pop eax
    iret
exc13 endp

; обработчик системного таймера
int_time proc uses eax

    ; получили текущее количество тиков
    mov  eax, cnt_time
    push eax
    ; вывели время
    mov edi, 80 * 2
    xor eax, eax
    test cnt_time, 05
    jz A
    test cnt_time, 09
    jnz skip

    mov al, ' '
```

```
    jmp pr
A:
    mov al, 'A'
pr:
    mov ah, 7
    stosw

skip:
    pop eax
; увеличили текущее количество счетчиков
    inc eax

    ; сохранили
    mov cnt_time, eax


    ; отправили EOI ведущему контроллеру прерываний
    mov         al, 20h
    out         20h, al
    ;pop eax
    iretd
int_time endp

int_keyboard 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

    ; mov ebx, 2 * 80 * 4
    ; call print_eax

    xor ebx, ebx
```

```
    mov bx, ax

    mov dl, ascii[ebx]
    mov ebx, syml_pos
    mov es:[ebx], dl

    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
int_keyboard endp

count_memory proc uses ds eax ebx
    mov ax, sel_DS_16bit
    mov ds, ax

    mov ebx, 100001h
    mov dl, 10101110b

    mov        ecx, 0FFEFFFFEh

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 ; 1 Mb
    div ebx

    mov ebx, 2 * 10
    call print_eax

    ret
count_memory endp

; вывод значения eax в видеобуффер
; в ebx позиция вывода на экран
print_eax proc uses ecx ebx edx
    add ebx, 10h
    mov ecx, 8

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], dl
    ror eax, 4
    sub ebx, 2
    loop print_symbol
    ret
print_eax endp

    pm_code_size = $-pm_start
```

```
pm_seg ends

rm_seg segment para public 'CODE' use16
    assume cs:rm_seg, ds:data_seg, ss: stack_seg

start:
    mov ax, data_seg
    mov ds, ax

    mov ax, pm_seg
    mov es, ax

    print_str msg_in_rm
    print_str msg_move_pm

    wait_key
    clear_screen

    xor         eax, eax

    ; загружаем адреса сегментов
    mov         ax, rm_seg
    load_gdt gdt_CS_16bit

    mov ax, pm_seg
    load_gdt gdt_CS_32bit

    mov ax, data_seg
    load_gdt gdt_DS_32bit

    mov ax, stack_seg
    load_gdt gdt_SS_32bit

    mov ax, data_seg
    shl eax, 4
    add         eax, offset gdt_null
    init_gdtr eax

    lea eax, es:dummy_exc
    load_idt trap_f
```

```
lea eax, es:dummy_exc
load_idt trap_s

lea eax, es:exc13
load_idt trap_13;

; загружаем дескриптор прерывания, нужно только смещение, тк селектор кода уже указан
lea eax, es:int_time
load_idt int08

lea eax, es:int_keyboard
load_idt int09

mov ax, data_seg
shl eax, 4
add        eax, offset IDT
init_idtr eax

; для возврата в защищенный:
;сохраним маски прерываний контроллеров
in        al, 21h
mov        mask_master, al
in        al, 0A1h
mov        mask_slave, al

; перепрограммируем пик (контроллер)
; вектор прерывания = базовый вектор прерывания + № IRQ
; irq0 - системный таймер, 8 + 0 = 8ое исключение => system fault, паника системы
; необходимо перепрограмировать пик на новый новый базовый вектор 32
set_interrupt_base 32

mov        al, 0FCh
out        21h, al
mov        al, 0FFh
out        0A1h, al

; загрузим IDT
lidt fword ptr idtr
```

```
    ; открытие линии A20
    in          al, 92h
    or          al, 2
    out          92h, al

    cli
    in          al, 70h
    or          al, 80h
    out          70h, al

    ; переход в защищенный режим
    mov          eax, cr0
    or eax, 1
    mov          cr0, eax

    db          66h
    far_jump pm_start, sel_CS_32bit


return_rm:

    ; возвращаемся в реальный режим
    mov          eax, cr0
    and          al, 0FEh
    mov          cr0, eax

    ; этот дальний переход необходим для модификации теневого регистра cs
    db          0EAh
    dw          $+4
    dw          rm_seg

    mov          eax, data_seg
    mov          ds, ax
    mov eax, pm_seg
    mov          es, ax
    mov          ax, stack_seg
    mov          ss, ax
    mov          ax, stack_size
    mov          sp, ax
```

```
    ;перепрограммируем контроллер
    set_interrupt_base 8

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

    ; загружаем таблицу дескриптров прерываний реального режима
    lidt         fword ptr idtr_backup

    in         al, 70h
    and         al, 7FH
    out         70h, al
    sti

    clear_screen
    print_str msg_out_pm

    mov         ax, 4C00h
    int         21h

    rm_code_size = $-start
rm_seg        ends
end start
```