Белорусский Государственный Университет Информатики и Радиоэлектроники

Кафедра ЭВМ

Отчет по лабораторной работе № 7
Тема: «Защищенный и реальный режим процессора. Переход из одного режима в другой и обработка прерываний»

Выполнил:
студент гр.950501 Поддубынй Д. П.

Проверил:
к.т.н., доцент Одинец Д.Н.

Минск 2021

1. Постановка задачи

Написать программу, которая выполняет следующие действия:

- Переход из реального режима в защищенный.

- Перехватывает аппаратное прерывание от клавиатуры, в обработчике которого считываются скан-коды клавиш и выводятся на экран. По нажатию клавиши «esc» осуществляется обратный переход в реальный режим.

- Перехватывает аппаратное прерывание от таймера, в обработчике которого отсчитывает секунды и выводит их на экран. По истечении времени, введенного при старте программы осуществляется обратный переход в реальный режим.

2. Алгоритм

1) Вводим время нахождения в защищенном режиме в секундах.
2) Настраиваем таймер на частоту 20 Гц.
3) Загружаем линейные адреса сегментов в дескрипторы.
4) Загружаем таблицу глобальных дескрипторов в регистр gdtr.
5) Запрещаем прерываний.
6) Сохраняем маски прерываний.
7) Инициализируем контроллеры.
8) Загружаем таблицу дескрипторов исключений в регистр idtr.
9) Переходим в защищенный режим.
10) Загружаем селекторы в регистры сегментов.
11) Разрешаем прерывания.
12) Производится обработка прерываний от клавиатуры и таймера.
13) Запрещаем прерываний.
14) Настраиваем теневые регистры сегментов для работы в реальном режиме.
15) Переходим в реальный режим.
16) Восстанавливаем значения сегментов.
17) Восстанавливаем таблицу векторов прерываний.
18) Инициализируем контроллеры.
19) Восстанавливаем маски прерываний.
20) Завершаемся.

3. *Листинг программы.*

```
.386P
.MODEL LARGE
;Структуры данных
S_DESC struc ;Структура сегментного дескриптора стр. 2881
LIMIT dw 0 ;Лимит сегмента (15:00)
BASE_L dw 0 ;Адрес базы, младшая часть (15:0)
BASE_M db 0 ;Адрес базы, средняя часть (23:16)
ACCESS db 0 ;Байт доступа
ATTRIBS db 0 ;Лимит сегмента (19:16) и атрибуты
BASE_H db 0 ;Адрес базы, старшая часть
S_DESC ends
I_DESC struc ;Структура дескриптора таблицы прерываний стр. 2983
OFFS_L dw 0 ;Адрес обработчика (0:15)
SEL dw 0 ;Селектор кода, содержащего код обработчика
PARAM_CNT db 0 ;Параметры
ACCESS db 0 ;Уровень доступа
OFFS_H dw 0 ;Адрес обработчика (31:16)
I_DESC ends
R_IDTR struc ;Структура IDTR стр. 2858
LIMIT dw 0
IDT_L dw 0 ;Смещение биты (0-15)
IDT_H dw 0 ;Смещение биты (31-16)
R_IDTR ends
;Флаги уровней доступа сегментов
ACS_PRESENT EQU 10000000B ;PXXXXXXX - бит присутствия, сегмент присутствует в оперативной памяти
ACS_CSEG EQU 00011000B ;XXXXIXXX - тип сегмента, для данных = 0, для кода 1
ACS_DSEG EQU 00010000B ;XXXSXXXX - бит сегмента, данный объект сегмент(системные объекты могут быть не сегменты)
ACS_READ EQU 00000010B ;XXXXXXRX - бит чтения, возможность чтения из другого сегмента
ACS_WRITE EQU 00000010B ;XXXXXXWX - бит записи, для сегмента данных разершает запись
ACS_CODE = ACS_PRESENT or ACS_CSEG ;AR сегмента кода
ACS_DATA = ACS_PRESENT or ACS_DSEG or ACS_WRITE;AR сегмента данных
ACS_STACK= ACS_PRESENT or ACS_DSEG or ACS_WRITE;AR сегмента стека
ACS_INT_GATE EQU 00001110B
ACS_TRAP_GATE EQU 00001111B ;XXXXSICR - сегмент, подчиненный сегмент кода, доступен для чтения
ACS_IDT EQU ACS_DATA ;AR таблицы IDT
ACS_INT EQU ACS_PRESENT or ACS_INT_GATE
ACS_TRAP EQU ACS_PRESENT or ACS_TRAP_GATE
ACS_DPL_3 EQU 01100000B ;X<DPL,DPL>XXXXX - привелегии доступа, доступ может получить любой код
```

```
;Сегмент кода реального режима
;*****************************  CODE  SEGMENT ****************************************
CODE_RM segment para use16 ;выравниваем по границе параграфов, используя 16-битные регистры
CODE_RM_BEGIN = $ ;адрес начала сегмента
assume cs:CODE_RM,DS:DATA,ES:DATA ;Инициализация регистров для ассемблирования
START:
mov ax,DATA ;Инициализиция сегментных регистров
mov ds,ax
mov es,ax
lea dx,MSG_ENTER
mov ah,9h
int 21h
call INPUT ;Ввод времени
mov ds:[TIME], al
call FILL_CR_0_BUFFER_RM
lea dx, BUFFER_CR_0_RM
mov ah, 9h
int 21h
lea dx,MSG_HELLO
mov ah,9h
int 21h
mov ah,7h
int 21h ;Ожидание подтверждения
PREPARE_RTC: ;Подготовка часов RTC
mov al,0Bh
out 70h,al ;Выбрать регистр состояния 0Bh
in al,71h ;Получить значение регистра 0Bh
or al,00000100b ;Установить бит DM в 1 - формат представления время в двоичном виде
out 71h,al ;Записать измененное значение
ENABLE_A20: ;Открыть линию A20
in al,92h
or al,2 ;Установить бит 1 в 1
out 92h,al
;Или так для старых компьютеров 0 LINE
;mov al, 0D1h
;out 64h, al
;mov al, 0DFh
;out 60h, al
SAVE_MASK: ;Сохранить маски прерываний
in al,21h
mov INT_MASK_M,al
in al,0A1h
mov INT_MASK_S,al
DISABLE_INTERRUPTS: ;Запрет маскируемых и немаскируемых прерываний
cli ;Запрет маскирумых прерываний
in al,70h
```

```
        or   al,10000000b  ;Установить  7  бит  в  1  для  запрета  немаскируемых
прерываний
        out 70h,al
        nop
        LOAD_GDT: ;Заполнить глобальную таблицу дескрипторов
        mov ax,DATA
        mov dl,ah
        xor dh,dh
        shl ax,4
        shr dx,4
        mov si,ax
        mov di,dx
        WRITE_GDT: ;Заполнить дескриптор GDT
        lea bx,GDT_GDT
        mov ax,si
        mov dx,di
        add ax,offset GDT
        adc dx,0
        mov [bx][S_DESC.BASE_L],ax
        mov [bx][S_DESC.BASE_M],dl
        mov [bx][S_DESC.BASE_H],dh
        WRITE_CODE_RM: ;Заполнить дескриптор сегмента кода реального режима
        lea bx,GDT_CODE_RM
        mov ax,cs
        xor dh,dh
        mov dl,ah
        shl ax,4
        shr dx,4
        mov [bx][S_DESC.BASE_L],ax
        mov [bx][S_DESC.BASE_M],dl
        mov [bx][S_DESC.BASE_H],dh
        WRITE_DATA: ;Записать дескриптор сегмента данных
        lea bx,GDT_DATA
        mov ax,si
        mov dx,di
        mov [bx][S_DESC.BASE_L],ax
        mov [bx][S_DESC.BASE_M],dl
        mov [bx][S_DESC.BASE_H],dh
        WRITE_STACK: ;Записать дескриптор сегмента стека
        lea bx, GDT_STACK
        mov ax,ss
        xor dh,dh
        mov dl,ah
        shl ax,4
        shr dx,4
        mov [bx][S_DESC.BASE_L],ax
        mov [bx][S_DESC.BASE_M],dl
        mov [bx][S_DESC.BASE_H],dh
        WRITE_CODE_PM: ;Записать дескриптор кода защищенного режима
        lea bx,GDT_CODE_PM
```

```
        mov ax,CODE_PM
        xor dh,dh
        mov dl,ah
        shl ax,4
        shr dx,4
        mov [bx][S_DESC.BASE_L],ax
        mov [bx][S_DESC.BASE_M],dl
        mov [bx][S_DESC.BASE_H],dh
        or [bx][S_DESC.ATTRIBS],40h
        WRITE_IDT: ;Записать дескриптор IDT
        lea bx,GDT_IDT
        mov ax,si
        mov dx,di
        add ax,OFFSET IDT
        adc dx,0
        mov [bx][S_DESC.BASE_L],ax
        mov [bx][S_DESC.BASE_M],dl
        mov [bx][S_DESC.BASE_H],dh
        mov IDTR.IDT_L,ax
        mov IDTR.IDT_H,dx
        FILL_IDT: ;Заполнить таблицу дескрипторов шлюзов прерываний
        irpc N, 0123456789ABCDEF ;Заполнить шлюзы 00-0F исключениями
        lea eax, EXC_0&N
        mov IDT_0&N.OFFS_L,ax
        shr eax, 16
        mov IDT_0&N.OFFS_H,ax
        endm
        irpc N, 0123456789ABCDEF ;Заполнить шлюзы 10-1F исключениями
        lea eax, EXC_1&N
        mov IDT_1&N.OFFS_L,ax
        shr eax, 16
        mov IDT_1&N.OFFS_H,ax
        endm
        lea eax, TIMER_HANDLER ;Поместить обработчик прерывания таймера на 20
шлюз
        mov IDT_TIMER.OFFS_L,ax
        shr eax, 16
        mov IDT_TIMER.OFFS_H,ax
        lea eax, KEYBOARD_HANDLER ;Поместить обработчик прерывания
клавиатуры на 21 шлюз
        mov IDT_KEYBOARD.OFFS_L,ax
        shr eax, 16
        mov IDT_KEYBOARD.OFFS_H,ax
        irpc N, 234567 ;Заполнить вектора 22-27 заглушками
        lea eax,DUMMY_IRQ_MASTER
        mov IDT_2&N.OFFS_L, AX
        shr eax,16
        mov IDT_2&N.OFFS_H, AX
        endm
        irpc N, 89ABCDEF ;Заполнить вектора 28-2F заглушками
```

```
        lea eax,DUMMY_IRQ_SLAVE
        mov IDT_2&N.OFFS_L,ax
        shr eax,16
        mov IDT_2&N.OFFS_H,ax
        endm
        lgdt fword ptr GDT_GDT ;Загрузить регистр GDTR
        lidt fword ptr IDTR ;Загрузить регистр IDTR
        mov eax,cr0 ;Получить управляющий регистр cr0
        or al,00000001b ;Установить бит PE в 1
        mov cr0,eax ;Записать измененный cr0 и тем самым включить защищенный режим
        OVERLOAD_CS: ;Перезагрузить сегмент кода на его дескриптор
        db 0EAH
        dw $+4
        dw CODE_RM_DESC
        OVERLOAD_SEGMENT_REGISTERS:      ;Переинициализировать      остальные сегментные регистры на дескрипторы
        mov ax,DATA_DESC
        mov ds,ax
        mov es,ax
        mov ax,STACK_DESC
        mov ss,ax
        xor ax,ax
        mov fs,ax ;Обнулить регистр fs
        mov gs,ax ;Обнулить регистр gs
        lldt  ax  ;Обнулить  регистр  LDTR  -  не  использовать  таблицы  локальных дескрипторов
        PREPARE_TO_RETURN:
        push cs ;Сегмент кода
        push offset BACK_TO_RM ;Смещение точки возврата
        lea edi,ENTER_PM ;Получить точку входа в защищенный режим
        mov eax,CODE_PM_DESC ;Получить дескриптор кода защищенного режима
        push eax ;Занести их в стек
        push edi
        REINITIALIAZE_CONTROLLER_FOR_PM:    ;Переинициализировать    контроллер прерываний на вектора 20h, 28h
        mov al,00010001b ;ICW1 - переинициализация контроллера прерываний
        out 20h,al ;Переинициализируем ведущий контроллер
        out 0A0h,al ;Переинициализируем ведомый контроллер
        mov al,20h ;ICW2 - номер базового вектора прерываний
        out 21h,al ;ведущего контроллера
        mov al,28h ;ICW2 - номер базового вектора прерываний
        out 0A1h,al ;ведомого контроллера
        mov al,04h ;ICW3 - ведущий контроллер подключен к 3 линии
        out 21h,al
        mov al,02h ;ICW3 - ведомый контроллер подключен к 3 линии
        out 0A1h,al
        mov  al,11h  ;ICW4  -  режим  специальной  полной  вложенности  для  ведущего контроллера
        out 21h,al
```

```
        mov  al,01h  ;ICW4 - режим обычной полной вложенности для ведомого контроллера
        out 0A1h,al
        mov al, 0 ;Размаскировать прерывания
        out 21h,al ;Ведущего контроллера
        out 0A1h,al ;Ведомого контроллера
        ENABLE_INTERRUPTS_0:    ;Разрешить маскируемые и немаскируемые прерывания
        in al,70h
        and  al,01111111b  ;Установить 7 бит в 0 для запрета немаскируемых прерываний
        out 70h,al
        nop
        sti ;Разрешить маскируемые прерывания
        GO_TO_CODE_PM: ;Переход к сегменту кода защищенного режима
        db 66h
        retf
        BACK_TO_RM: ;Точка возврата в реальный режим
        cli ;Запрет маскируемых прерываний
        in al,70h ;И не маскируемых прерываний
        or  AL,10000000b  ;Установить 7 бит в 1 для запрета немаскируемых прерываний
        out 70h,AL
        nop
        REINITIALISE_CONTROLLER: ;Переинициализация контроллера прерываний
        mov al,00010001b ;ICW1 - переинициализация контроллера прерываний
        out 20h,al ;Переинициализируем ведущий контроллер
        out 0A0h,al ;Переинициализируем ведомый контроллер
        mov al,8h ;ICW2 - номер базового вектора прерываний
        out 21h,al ;ведущего контроллера
        mov al,70h ;ICW2 - номер базового вектора прерываний
        out 0A1h,al ;ведомого контроллера
        mov al,04h ;ICW3 - ведущий контроллер подключен к 3 линии
        out 21h,al
        mov al,02h ;ICW3 - ведомый контроллер подключен к 3 линии
        out 0A1h,al
        mov  al,11h ;ICW4 - режим специальной полной вложенности для ведущего контроллера
        out 21h,al
        mov  al,01h  ;ICW4 - режим обычной полной вложенности для ведомого контроллера
        out 0A1h,al
        PREPARE_SEGMENTS:  ;Подготовка сегментных регистров для возврата в реальный режим
        mov GDT_CODE_RM.LIMIT,0FFFFh ;Установка лимита сегмента кода в 64KB
        mov GDT_DATA.LIMIT,0FFFFh ;Установка лимита сегмента данных в 64KB
        mov GDT_STACK.LIMIT,0FFFFh ;Установка лимита сегмента стека в 64KB
        db 0EAH ;Перезагрузить регистр cs
        dw $+4
        dw CODE_RM_DESC ;На сегмент кода реального режима
```

```
        mov ax,DATA_DESC ;Загрузим сегментные регистры дескриптором сегмента данных
        mov ds,ax
        mov es,ax
        mov fs,ax
        mov gs,ax
        mov ax,STACK_DESC
        mov ss,ax ;Загрузим регистр стека дескриптором стека
        ENABLE_REAL_MODE: ;Включим реальный режим
        mov eax,cr0
        and al,11111110b ;Обнулим 0 бит регистра cr0
        mov cr0,eax
        db 0EAH
        dw $+4
        dw CODE_RM ;Перезагрузим регистр кода
        mov ax,STACK_A
        mov ss,ax
        mov ax,DATA
        mov ds,ax
        mov es,ax
        xor ax,ax
        mov fs,ax
        mov gs,ax
        mov IDTR.LIMIT, 3FFH
        mov dword ptr IDTR+2, 0
        lidt fword ptr IDTR
        REPEAIR_MASK: ;Восстановить маски прерываний
        mov al,INT_MASK_M
        out 21h,al ;Ведущего контроллера
        mov al,INT_MASK_S
        out 0A1h,al ;Ведомого контроллера
        ENABLE_INTERRUPTS:    ;Разрешить    маскируемые    и    немаскируемые прерывания
        in al,70h
        and  al,01111111b  ;Установить  7  бит  в  0  для  разрешения  немаскируемых прерываний
        out 70h,al
        nop
        sti ;Разрешить маскируемые прерывания
        DISABLE_A20: ;Закрыть вентиль A20
        in al,92h
        and al,11111101b ;Обнулить 1 бит - запретить линию A20
        out 92h, al
        EXIT: ;Выход из программы
        mov ax,3h
        int 10H ;Очистить видео-режим
        lea dx,MSG_EXIT
        mov ah,9h
        int 21h ;Вывести сообщение
```

```
call FILL_CR_0_BUFFER_RM
lea dx, BUFFER_CR_0_RM
mov ah, 9h
int 21h
mov ax,4C00h
int 21H ;Выход в dos


INPUT proc near ;Процедура ввода время-нахождения в защищенном режиме
mov ah,0ah
xor di,di
mov dx,offset ds:[INPUT_TIME]
int 21h
mov dl,0ah
mov ah,02
int 21h
mov si,offset INPUT_TIME+2
cmp byte ptr [si],"-"
jnz ii1
mov di,1
inc si
II1:
xor ax,ax
mov bx,10
II2:
mov cl,[si]
cmp cl,0dh
jz ii3
cmp cl,'0'
jl er
cmp cl,'9'
ja er
sub cl,'0'
mul bx
add ax,cx
inc si
jmp ii2
ER:
mov dx, offset MSG_ERROR
mov ah,09
int 21h
int 20h
II3:
ret
INPUT endp

FILL_CR_0_BUFFER_RM proc near
push eax
push esi
push dx
```

```
mov eax, cr0
xor dx, dx
mov cx, 32
lea esi, BUFFER_CR_0_RM
fill_cr_0_loop_rm:
mov dl, al
shl dl, 7
shr dl, 7
shr eax, 1
add dl, 48
mov [esi], dl
inc esi
xor dl, dl
loop fill_cr_0_loop_rm

pop dx
pop esi
pop eax
ret
FILL_CR_0_BUFFER_RM endp

SIZE_CODE_RM = ($ - CODE_RM_BEGIN) ;Лимит сегмента кода

CODE_RM ends



;Сегмент кода реального режима
CODE_PM segment para use32
CODE_PM_BEGIN = $
assume cs:CODE_PM,ds:DATA,es:DATA ;Указание сегментов для компиляции
ENTER_PM: ;Точка входа в защищенный режим
call CLRSCR ;Процедура очистки экрана
xor edi,edi ;В edi смещение на экране
lea esi,MSG_HELLO_PM ;В esi адрес буфера
call BUFFER_OUTPUT ;Вывести строку-приветствие в защищенном режиме
add edi,160 ;Перевести курсор на следующую строку
lea esi,MSG_KEYBOARD
call BUFFER_OUTPUT ;Вывести поле для вывода скан-кода клавиатуры
mov edi,320
lea esi,MSG_TIME
call BUFFER_OUTPUT ;Вывести поле для вывода время
mov edi,480
lea esi,MSG_COUNT
call BUFFER_OUTPUT

call FILL_CR_0_BUFFER
mov edi, 640
lea esi, BUFFER_CR_0
```

```
call BUFFER_OUTPUT

mov DS:[COUNT],0
WAITING_ESC: ;Ожидание нажатия кнопки выхода из защищенного режима
jmp WAITING_ESC ;Если был нажат не ESC
EXIT_PM: ;Точка выхода из 32-битного сегмента кода
db 66H
retf ;Переход в 16-битный сегмент кода
EXIT_FROM_INTERRUPT: ;Точка выхода для выхода напрямую из обработчика прерываний
popad
pop es
pop ds
pop eax ;Снять со стека старый EIP
pop eax ;CS
pop eax ;И EFLAGS
sti ;Обязательно, без этого обработка аппаратных прерываний отключена
db 66H
retf ;Переход в 16-битный сегмент кода


; ---------------------------------------------------------------------------------------------
WORD_TO_DEC proc near ;Процедура перевода слова в строку
pushad
movzx eax,ax
xor cx,cx
mov bx,10
LOOP1: ;Цикл по извлечению цифры
xor dx,dx
div bx
add dl,'0'
push dx
inc cx
test ax,ax
jnz LOOP1
LOOP2: ;Цикл по заполнению буфера
pop dx
mov [di],dl
inc di
loop LOOP2
popad
ret
WORD_TO_DEC endp
; ---------------------------------------------------------------------------------------------

FILL_CR_0_BUFFER proc near
push eax
push esi
push dx
```

```
        mov eax, cr0
        xor dx, dx
        mov cx, 32
        lea esi, BUFFER_CR_0
        fill_cr_0_loop:
        mov dl, al
        shl dl, 7
        shr dl, 7
        shr eax, 1
        add dl, 48
        mov [esi], dl
        inc esi
        xor dl, dl
        loop fill_cr_0_loop

        pop dx
        pop esi
        pop eax
        ret
        FILL_CR_0_BUFFER endp


        ; ---------------------------------------------------------------------------------------------
        DIGIT_TO_HEX  proc  near ;Процедура перевода цифры в шеснадцатеричный
вид
        add al,'0'
        cmp al,'9'
        jle DTH_END
        add al,7
        DTH_END:
        ret
        DIGIT_TO_HEX endp
        ; ---------------------------------------------------------------------------------------------


        ; ---------------------------------------------------------------------------------------------
        BYTE_TO_HEX  proc  near ;Процедура перевода числа в шеснадцатеричный
вид
        push ax
        mov ah,al
        shr al,4
        call DIGIT_TO_HEX
        mov [di],al
        inc di
        mov al,ah
        and al,0Fh
        call DIGIT_TO_HEX
        mov [di],al
        inc di
        pop ax
```

```
        ret
        BYTE_TO_HEX endp
        ; ---------------------------------------------------------------------------------------------


        M = 0
        IRPC N, 0123456789ABCDEF
        EXC_0&N label word ;Обработчики исключений
        cli
        jmp EXC_HANDLER
        endm
        M = 010H
        IRPC N, 0123456789ABCDEF ;Обработчики исключений
        EXC_1&N label word
        cli
        jmp EXC_HANDLER
        endm

        ; ---------------------------------------------------------------------------------------------
        EXC_HANDLER proc near ;Процедура вывода обработки исключений
        call CLRSCR ;Очистка экрана
        lea esi, MSG_EXC
        mov edi, 40*2
        call BUFFER_OUTPUT ;Вывод предупреждения
        pop eax ;Снять со стека старый EIP
        pop eax ;CS
        pop eax ;И EFLAGS
        sti ;Обязательно, без этого обработка аппаратных прерываний отключена
        db 66H
        retf ;Переход в 16-битный сегмент кода
        EXC_HANDLER ENDP
        ; ---------------------------------------------------------------------------------------------

        ; ---------------------------------------------------------------------------------------------
        DUMMY_IRQ_MASTER proc near ;Заглушка для аппаратных прерываний
ведущего контроллера
        push eax
        mov al,20h
        out 20h,al
        pop eax
        iretd
        DUMMY_IRQ_MASTER endp
        ; ---------------------------------------------------------------------------------------------

        ; ---------------------------------------------------------------------------------------------
        DUMMY_IRQ_SLAVE proc near ;Заглушка для аппаратных прерываний
ведомого контроллера
        push eax
        mov al,20h
        out 20h,al
```

```
out 0A0h,al
pop eax
iretd
DUMMY_IRQ_SLAVE endp
; ---------------------------------------------------------------------------------------------------

; ---------------------------------------------------------------------------------------------------
TIMER_HANDLER proc near ;Обработчик прерываний системного таймера
push ds
push es
pushad ;Занести в стек расширенные регистры общего назначения
mov ax,DATA_DESC ;Переинициализировать сегментные регистры
mov ds,ax
inc ds:[COUNT] ;Увеличить значение счетчика
lea edi,ds:[BUFFER_COUNT]
mov ax,ds:[COUNT]
call WORD_TO_DEC ;Преобразовать счётчик в строку
mov edi,538
lea esi,BUFFER_COUNT
call BUFFER_OUTPUT ;Вывести значение счетчика
SHOW_TIMER:
mov al,0h ;Выбрать регистр секунд cmos
out 70h,al
in al,71h ;Прочитать значение секунд
cmp al,ds:[SECOND] ;Если секунда та же самая
je SKIP_SECOND ;То пропустить вывод
mov ds:[SECOND],al ;Иначе записать значение новой секунды
mov al,ds:[TIME] ;Получить значение оставшегося время
cmp ds:[TIME],0 ;Если время подошло к концу
je DISABLE_PM ;То на выход из защищенного режима
xor ah,ah
lea edi,ds:[BUFFER_TIME]
call WORD_TO_DEC ;Преобразовать его в строку
mov edi,356
lea esi,BUFFER_TIME
call BUFFER_OUTPUT ;Вывести значение оставшегося время
dec ds:[TIME] ;Уменьшить значение оставшегося времени
lea esi,BUFFER_TIME
call BUFFER_CLEAR ;Очистка буфера
jmp SKIP_SECOND ;На выход из обработки время
DISABLE_PM: ;Выход из защищенного режима
mov al,20h
out 20h,al
db 0eah ;Дальний jmp
dd OFFSET EXIT_FROM_INTERRUPT ;На метку
dw CODE_PM_DESC ;В сегменте
SKIP_SECOND: ;Секунда та же, не надо производить никаких действий
mov al,20h
out 20h,al ;Отпарвка сигнала контроллеру прерываний
popad
```

```
pop es
pop ds
iretd
TIMER_HANDLER endp
; ---------------------------------------------------------------------------------------------

; ---------------------------------------------------------------------------------------------
KEYBOARD_HANDLER proc near ;Обработчик прерывания клавиатуры
push ds
push es
pushad ;Сохранить расширенные регистры общего назначения
in al,60h ;Считать скан код последней нажатой клавиши ;
cmp al,1 ;Если был нажат 'ESC'
je KEYBOARD_EXIT ;Тогда на выход из защищенного режима
mov ds:[KEY_SCAN_CODE],al ;Записать его в память
lea edi,ds:[BUFFER_SCAN_CODE]
mov al,ds:[KEY_SCAN_CODE]
xor ah,ah
call BYTE_TO_HEX ;Преобразовать скан-код в строку
mov edi,200
lea esi,BUFFER_SCAN_CODE
call BUFFER_OUTPUT ;Вывести строку со скан-кодом
jmp KEYBOARD_RETURN
KEYBOARD_EXIT:
mov al,20h
out 20h,al
db 0eah
dd OFFSET EXIT_FROM_INTERRUPT
dw CODE_PM_DESC
KEYBOARD_RETURN:
mov al,20h
out 20h,al ;Отпарвка сигнала контроллеру прерываний
popad ;Восстановить значения регистров
pop es
pop ds
iretd ;Выход из прерывания
KEYBOARD_HANDLER endp
; ---------------------------------------------------------------------------------------------

; ---------------------------------------------------------------------------------------------
CLRSCR proc near ;Процедура очистки консоли
push es
pushad
mov ax,TEXT_DESC ;Поместить в ax дескриптор текста
mov es,ax
xor edi,edi
mov ecx,80*25 ;Количество символов в окне
mov ax,700h
rep stosw
popad
```

```
pop es
ret
CLRSCR endp
; ---------------------------------------------------------------------------------------------

; ---------------------------------------------------------------------------------------------
BUFFER_CLEAR proc near ;Процедура очистки буфера
mov al,' '
mov [esi+0],al
mov [esi+1],al
mov [esi+2],al
mov [esi+3],al
mov [esi+4],al
mov [esi+5],al
mov [esi+6],al
mov [esi+7],al
ret
BUFFER_CLEAR endp
; ---------------------------------------------------------------------------------------------

; ---------------------------------------------------------------------------------------------
BUFFER_OUTPUT proc near ;Процедура вывода текстового буфера,
оканчивающегося 0
push es
PUSHAD
mov ax,TEXT_DESC ;Поместить в es селектор текста
mov es,ax
OUTPUT_LOOP: ;Цикл по выводу буфера
lodsb
or al,al
jz OUTPUT_EXIT ;Если дошло до 0, то конец выхода
stosb
inc edi
jmp OUTPUT_LOOP
OUTPUT_EXIT: ;Выход из процедуры вывода
popad
pop es
ret
BUFFER_OUTPUT ENDP
; ---------------------------------------------------------------------------------------------


SIZE_CODE_PM = ($ - CODE_PM_BEGIN)
CODE_PM ENDS


;Сегмент данных реального/защищенного режима
;*************************** DATA SEGMENT
**************************************************8
DATA segment para use16 ;Сегмент данных реального/защищенного режима
```

```
DATA_BEGIN = $
;GDT - глобальная таблица дескрипторов
;_____________________________________________________________________
GDT_BEGIN = $
GDT label word ;Метка начала GDT (GDT: не работает)
GDT_0 S_DESC <0,0,0,0,0,0>
GDT_GDT S_DESC <GDT_SIZE-1,,,ACS_DATA,0,>
GDT_CODE_RM S_DESC <SIZE_CODE_RM-1,,,ACS_CODE,0,>
GDT_DATA S_DESC <SIZE_DATA-1,,,ACS_DATA+ACS_DPL_3,0,>
GDT_STACK S_DESC <1000h-1,,,ACS_DATA,0,>
GDT_TEXT S_DESC <2000h-1,8000h,0Bh,ACS_DATA+ACS_DPL_3,0,0>
GDT_CODE_PM S_DESC <SIZE_CODE_PM-1,,,ACS_CODE+ACS_READ,0,>
GDT_IDT S_DESC <SIZE_IDT-1,,,ACS_IDT,0,>
GDT_SIZE = ($ - GDT_BEGIN) ;Размер GDT
;_____________________________________________________________________|

;Селлекторы сегментов
CODE_RM_DESC = (GDT_CODE_RM - GDT_0)
DATA_DESC = (GDT_DATA - GDT_0)
STACK_DESC = (GDT_STACK - GDT_0)
TEXT_DESC = (GDT_TEXT - GDT_0)
CODE_PM_DESC = (GDT_CODE_PM - GDT_0)
IDT_DESC = (GDT_IDT - GDT_0)

;IDT - таблица дескрипторов прерываний
;_____________________________________________________________________
IDTR R_IDTR <SIZE_IDT,0,0> ;Формат регистра ITDR
IDT label word ;Метка начала IDT
IDT_BEGIN = $
IRPC N, 0123456789ABCDEF
IDT_0&N I_DESC <0, CODE_PM_DESC,0,ACS_TRAP,0> ; 00...0F
ENDM
IRPC N, 0123456789ABCDEF
IDT_1&N I_DESC <0, CODE_PM_DESC, 0, ACS_TRAP, 0> ; 10...1F
ENDM
IDT_TIMER   I_DESC   <0,CODE_PM_DESC,0,ACS_INT,0>   ;IRQ   0   -   прерывание системного таймера
IDT_KEYBOARD  I_DESC  <0,CODE_PM_DESC,0,ACS_INT,0>  ;IRQ  1  -  прерывание клавиатуры
IRPC N, 23456789ABCDEF
IDT_2&N I_DESC <0, CODE_PM_DESC, 0, ACS_INT, 0> ; 22...2F
ENDM
SIZE_IDT = ($ - IDT_BEGIN) ; Размер IDT
;_____________________________________________________________________|


MSG_HELLO db "press any key to go to the protected mode",13,10,"$"
MSG_HELLO_PM db "wellcome in protected mode, press esc to come back to the real mode",0
MSG_EXIT db "wellcome back to real mode",13,10,"$"
```

```
        MSG_KEYBOARD db "keyboard scan code:",0
        MSG_TIME db "go back to RM in XXXXXXX seconds",0
        MSG_COUNT db "quantity of interrupt calls:",0
        MSG_EXC db "exception: XX",0
        MSG_ENTER db "enter time in protected mode: $"
        MSG_ERROR db "incorrect error$"
        HEX_TAB db "0123456789ABCDEF" ;Таблица номеров исключений
        ESP32 dd 1 dup(?) ;Указатель на вершину стека
        INT_MASK_M db 1 dup(?) ;Значение регистра масок ведущего контроллера
        INT_MASK_S db 1 dup(?) ;Значение регистра масок ведомого контроллера
        KEY_SCAN_CODE db 1 dup(?) ;Ска-код последней нажатой клавиши
        SECOND db 1 dup(?) ;Текущее значение секунд
        TIME db 1 dup(10) ;Время нахождения в зазищенном режиме
        COUNT dw 1 dup(0) ;Количество вызовов прерывания (диапазон от 0 до
65535)
        BUFFER_COUNT db 8 dup(' ') ;Буфер для вывода количества вызовов
прерываинй
        db 1 dup(0)
        BUFFER_SCAN_CODE db 8 dup(' ') ;Буфер для вывода скан-кода клавиатуры
        db 1 dup(0)
        BUFFER_TIME db 8 dup(' ') ;Буфер для вывода оставшегося время в
защищенном режиме
        db 1 dup(0)
        INPUT_TIME db 6,7 dup(?) ;Буфер для ввода время
        BUFFER_CR_0 db 32 dup('?')
        db 1 dup(0)
        BUFFER_CR_0_RM db 32 dup('?'), 13, 10, "$"
        SIZE_DATA = ($ - DATA_BEGIN) ;Размер сегмента данных
        DATA ends
        ;Сегмент стека реального/защищенного режима
        STACK_A segment para stack
        db 1000h dup(?)
        STACK_A ends
        end START
```

4. Результаты работы программы.